THE UNIVERSITY OF
WARWICK

AUTHOR: **Nicolai Peremezhney** DEGREE: **Ph.D.**

TITLE: **Chemical product/process design and optimization: development of novel techniques and integration of bio-feedstocks.**

DATE OF DEPOSIT: .....

I agree that this thesis shall be available in accordance with the regulations governing the University of Warwick theses.

I agree that the summary of this thesis may be submitted for publication.

I **agree** that the thesis may be photocopied (single copies for study purposes only).

Theses with no restriction on photocopying will also be made available to the British Library for microfilming. The British Library may supply copies to individuals or libraries. subject to a statement from them that the copy is supplied for non-publishing purposes. All copies supplied by the British Library will carry the following statement:

> "Attention is drawn to the fact that the copyright of this thesis rests with its author. This copy of the thesis has been supplied on the condition that anyone who consults it is understood to recognise that its copyright rests with its author and that no quotation from the thesis and no information derived from it may be published without the author's written consent."

AUTHOR'S SIGNATURE: .....

USER'S DECLARATION

1. I undertake not to quote or make use of any information from this thesis without making acknowledgement to the author.
2. I further undertake to allow no-one else to use this thesis while it is in my care.

DATE SIGNATURE ADDRESS

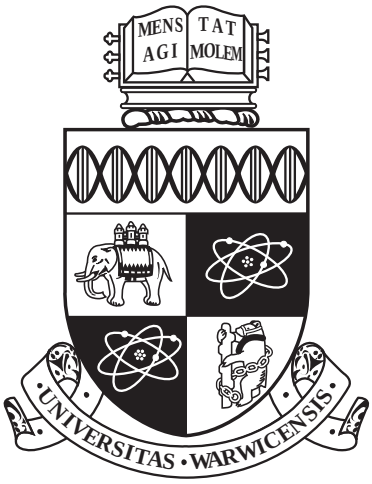
.....

.....

.....

.....

.....



Chemical product/process design and optimization: development of novel techniques and integration of bio-feedstocks.

by

Nicolai Peremezhney

Thesis

Submitted to the University of Warwick

for the degree of

Doctor of Philosophy

Complexity

December 2013

THE UNIVERSITY OF
WARWICK

Contents

Acknowledgments

There are several people to whom I wish to express my gratitude. I would like to thank Professor Alexei Lapkin for his great supervision, support and encouragement during this work. Your advice on both research as well as on my career have been extremely useful. I would also like to thank Associate Professor Colm Connaughton for being a great adviser. I would like to thank you for encouraging my research and for allowing me to grow as a research scientist. I would like to express my appreciation to Professor Evor Hines for fruitful discussions during my work. My gratitude is extended to all members of the Reaction Engineering group at Cambridge University for their help and support. Last but not least I would like to express the warmest thank you to my girlfriend, Emma, for always backing me up and encouraging me to struggle on.

Declarations

I herewith declare that this thesis contains my own research performed under the supervision of Professor Alexei Lapkin, without assistance of third parties. No part of this thesis was previously submitted for a degree at any other university.

Material in chapter 2 is based on:
Peremezhney, N., Connaughton, C., Hines, E., Lapkin, A.A. (2013). Combining Gaussian Processes, Mutual Information and a Genetic Algorithm for multi-target optimization of expensive-to-evaluate functions. *Engineering Optimization.* (Accepted for publication).

Material in chapter 4 is based on:
Peremezhney, N., Connaughton, C., Unali, G., Hines, E., Lapkin, A.A. (2012). Application of dimensionality reduction to visualisation of high-throughput data and building of a classification model in formulated consumer product design. *Chemical Engineering Research and Design*, **90**(12), pp. 2179-2185.

Abstract

The research topic addressed in this thesis is the development of new ideas and techniques for acceleration and automation of processes involved in the design of chemical products with predefined properties. In particular, we demonstrate techniques that address the shortcomings of the existing methods and take a bird's-eye view over the new possible directions for chemical product development necessitated by the integration of bio-feedstocks into the existing supply chain. Furthermore, we introduce an approach for sequential, on-line multi-target product/process optimization in a scenario where: automation of the overall design process is sought; adequate physical models are not available; unknown constraints on the decision space may be present; and resources are limited or costly. We test the approach on a number of simulations. The results indicate that the approach is able to, in a modest number of iterations, find solutions associated with the targets to a satisfactory degree of accuracy. In addition, for supervised problems where categorical data are available, we introduce an approach that allows one to perform categorization of a given product composition according to a particular property. We test our solutions empirically on real data. The results show that the approach compares well with existing state of the art techniques. We also investigate the application of a variety of nonlinear dimensionality techniques to the visualisation of chemical product data.

Chapter 1

Introduction

The aim of chemical product design is to find a product that exhibits certain specified behaviour/properties, corresponding to the desired functional properties. Thus, in the area of chemical consumer products (sometimes termed formulated products) the main useful functions, for example the function of 'moisturising' or the function of 'UV protection', are achieved through the use of molecules and particles with the corresponding required physico-chemical properties, e.g., UV absorbance and scattering of light by TiO_2 particles, thus reducing the flux of the harmful range of UV solar radiation to the skin. These main useful functions in consumer products are accompanied by a varied cocktail of secondary desired functions, such as 'feel', 'smell', 'colour', etc. Various harmful functions could also be specified, which include any side effects, the cost to the consumer, the cost to the manufacture and the environmental impact of manufacture and use of the products. Traditionally, success of the developed chemical consumer product relied on an engineer's [formulator's] experiential knowledge of what combinations of ingredients (formulations) work for the required product application/s as well as the engineer being able to, through random experimentation or via the use of statistical/machine learning techniques, find the composition of proportions of the identified ingredients that translate into a product with predefined *values/qualitative descriptions* of the desired main and secondary functional properties, e.g. transparency of 80%, high viscosity liquid, etc.

The demand for 'green' chemical products coupled with strong competition in the market place, dictating shorter product-to-market times, is prompting the manufacturers to update their ingredient libraries with molecules not previously used and to examine the suitability of the existing approaches to product design. Currently, a significant amount of research is going into the development of data driven ap-

proaches, based on mechanistic or empirical modelling, aimed at helping reduce the costs associated with product development. The areas under scrutiny are: experimental design, process/predictive modelling, and optimization.

The needs of chemical product design dictate that the data driven methods involved deliver:

- Process models, physical or empirical, that are: capable of characterizing the effects of the interplay among product constituents on product properties.
- Design of experiments that reveal the most information about the underlying physical process.
- Target/global optimization approaches that can deal with multiple objectives.

1.1 Observations with regards to current state of affairs

1. Physical modelling involves constructing models based on the underlying physics and chemistry governing the behaviour of the process. It bears high costs in terms of human effort and expertise, and can also be very time consuming. It is, thus, rarely employed. Empirical modelling is carried out by considering only the functions that belong to a restricted class or a number of classes, which may result in processes not being well modelled and, consequently, in unsatisfactory approximations. Moreover, as experimental design is performed in conjunction with the model that is to be fitted to the data, considering only a particular model class or classes bears direct consequence on the quality of experimental designs and, subsequently, on the outcome of optimization.
2. There is a lack of data driven techniques addressing the scenario of multi-target optimization in the presence of limited amount and/or high cost of materials.
3. There is a lack of real time multi-target optimization approaches where modelling, experimentation and optimization are part of a closed feed back based loop.

1.1.1 Discussion of observation (1)

Successful design and optimization of a chemical product depends on the availability of adequate models relating composition of the product constituents to the product's properties. Physical models are usually the preferred choice, but are often

unavailable. Research and development efforts of chemical consumer product manufacturers, however, are seldom concentrated to address this issue. The reason for it is that development of fundamental knowledge, being time consuming and expensive in term of human expertise, is seldom justified, as research and development costs are greatly influenced by a product's short lifespan on the market [1]. The prevalent approach in the industry is to employ empirical modelling through the use of Design Of Experiments (DOE) together with Response Surface Methodology (RSM). DOE is a procedure for preparing experiments so that the data collected can be analysed to bring about valid conclusions. When chosen well, experimental designs maximize the amount of 'information' that can be obtained with the planned experiments. RSM is a group of mathematical/statistical techniques for building empirical models. The objectives are, using a chosen DOE, to optimize the output variable (response), which is affected by a number of input variables, and to create a predictive model of the relationship between the input variables and the response. A typical sequence of steps for an approach based on joint application of DOE and RSM is:

1. Select a statistical model to approximate the response surface with and an optimization algorithm to drive the search for the optimum solution.

2. Given the model, employ a DOE method to choose a set of input configurations for which the values of the output variable are to be obtained via interaction with the real system - the training set.

3. Use the training set to train and validate (through cross-validation) the statistical model.

4. Use the trained statistical model together with a chosen optimization algorithm to predict the optimum solution.

The RSM-DOE based approaches are often made sequential by repeating steps 2 to 4 until a chosen stopping criteria is met. The training data set size is usually of the order 10 times the number of the input variables. As obtaining a separate test data set is often expensive, k-fold cross-validation is performed at model validation stage with the value of k chosen in the range between 10 and N (leave-one-out), where N is the size of the available dataset. Models for response surface approximation, presently most favoured by the industry, are Polynomial Regression and Artificial Neural Networks (ANNs). The main draw-back of employing polynomial approximation is that modelling is carried out by considering only the functions that belong to a restricted class or a number of classes, which may result in the process

not being well modelled and, consequently, in unsatisfactory approximations. Some of the biggest shortcomings of ANNs are the lack of methodology for choosing the number of hidden layers (and the number of units in a hidden layer) and difficulties associated with relating network weights to physical parameters [2]. In the work presented in this thesis we apply Gaussian Process Regression (GPR) [35] for response surface approximation. This probabilistic non-parametric technique is truly data driven, as it seeks the relationship among the observations, rather than trying to approximate the modelled process by fitting the parameters of the selected basis functions [4]. GPR based optimisation approaches have become a very popular choice for global optimisation problems. However, research into theoretical performance of GPR based optimization is still in it's infancy [36]. Although RSM based techniques are currently a popular choice, non-model based direct search optimisation techniques such as the NelderMead simplex algorithm [13], for instance, have also, historically, been employed in the industry.

1.1.2 Discussion of observation (2)

A frequently occurring situation in chemical product design is that of the limited amount or high cost of materials. In this scenario it is possible that the training set obtained, whilst providing enough information to train a good predictive model, may contain a high proportion of points that are not near the optimum, which would lead to inefficient use of resources. Exploration of approaches that deal with the scenario of expensive function evaluations has recently received significant attention in the research community. A number of approaches have been put forward that have a common underlying scheme, namely, the idea of integrating the DOE into optimization [30; 33; 36; 7; 8; 37; 5; 6; 9; 10]. The key steps of such an approach are:

1. Sample a small proportion of the allowable number of training points and train the predictive model.

2. Approximate the response surface, select a solution that satisfies both the criteria for optimality of the DOE and the optimization problem in hand as a whole. Evaluate the solution. If the optimum solution has been found, stop. Otherwise, go to step 3.

3. Augment the training set, update the model and go to step 2.

GPR is particularly well suited for such sequential optimization. The variance of the predicted observations can be used to highlight ill-explored regions of the design

space, whilst the mean predictions could be employed to give estimates of proximity to the target or a measure of improvement on the current optimum if the optimization is global. Variants of a single-objective, sequential response surface model based optimization approach have already been tried in the field of chemical reaction optimization. In [11], for instance, maximization of the *trans*-stilbene conversion rate in the epoxidation of trans-stilbene over Co^{2+}-NaX catalyst is performed to validate a sequential optimization approach based on RSM. The authors utilise Gaussian Processes as the model for response surface approximation. At each iteration the mean and variance of the predicted observations are checked against a predefined pair of mean and variance values through a set of inequalities, thus guiding the search for the regions of the design space to be sampled in the next batch of experiments. Techniques for on-line sequential *multi-target* optimization of chemical products/processes in the presence of limited amount and/or high cost of materials, to our knowledge, are not currently available. In this thesis we present a technique for such optimization.

1.1.3 Discussion of observation (3)

At present, a hot research topic is the automation of chemical product/process development. To this end, there has been some success within the development of 'closed-loop' micro-reactor systems that incorporate real-time experimentation, statistical modelling and sequential DOE. By integrating sensors capable of reading off the output measurements and a computer program, running a machine learning algorithm/s that accepts the sensor data and outputs information about a suggested next experiment, within a continuous flow reactor, the automation of the overall reaction optimization is achieved. In [12] for instance, the authors present a self-optimising micro-reactor system that employs the NelderMead Simplex algorithm for optimization of a Heck Reaction. To our knowledge, an on-line sequential 'closed-loop' feedback integrated approach has not yet been applied for *multi-target* chemical product/process optimization.

1.2 Categorisation of product compositions (formulations) and data visualisation

As well as exhibiting specified behaviour, corresponding to the desired main functional properties, a chemical consumer product has to be able to perform a number of secondary, customer dictated desired functions, such as 'feel', 'smell', 'colour', etc.

Performance criteria for such functions is not easily quantifiable. In such instances the output domain is no longer a set of continuous values, as in a regression case, but rather a set of categories, representing the spectrum of qualitative descriptions of the desired secondary function. The task is then to first, design and evaluate a set of experiments most informative about the location of the category boundary/ies, then, select and train a classification model capable of robustly predicting the category of the desired function, given a particular formulation.

The kind of problems encountered in chemical product design are often characterised by high dimensionality. Applying classification in these instances can cause biased estimates [3], thus affecting the accuracy of predictions. It can also lead to high costs in terms of computation time [14]. To address these issues, in this thesis we perform classification in the reduced space via application of a dimensionality reduction technique. There are a number and a variety of techniques available for dimensionality reduction. The widely known and used ones are Principal Component Analysis (PCA)[61] and multidimensional scaling (MDS)[62]. However, the effectiveness of both PCA and MDS is limited by their global linearity. In order to resolve the problem of dimensionality reduction in nonlinear cases, the manifold learning techniques such as locally linear embedding (LLE) [64] and Isomap [65], for instance, could be employed. Classification in the reduced space could be performed by a simple technique such as a k-NN [72], for instance. It must be noted, however, that for some highly non-linear problems where class separation is difficult (in the reduced space) it may be more appropriate to employ a probabilistic model such as multinomial logistic regression or Gaussian process classification (GPC) [35] or integrate fuzzy logic [60] into classification.

In this thesis we perform classification in the reduced space provided by a non-linear supervised dimensionality reduction technique such as SIsomap [66]. To gain actionable insight, it is also important that the available evaluated data are visualised. In this respect, unless the number of ingredients in a product is not higher than three, it is also necessary to consider the employment of a dimensionality reduction technique.

In this thesis we introduce an approach that incorporates dimensionality reduction as part of an algorithm for categorization of a given product composition according to a particular property. We compare prediction results of this approach with several well-established classification models. We also investigate the application of a

variety of non-linear dimensionality techniques to visualisation of chemical product data.

1.3 Integration of bio-feedstocks and new possible directions for consumer product development

At present the use of bio-feedstocks in consumer product design is relatively limited, due to a small range of molecules available on the market, primarily natural oils, flavour and fragrance substances, nutraceuticals, bio-pharmaceuticals. However, this range is expected to be rapidly expanded, offering new opportunities for product design. The emerging question is whether our existing methods of product design in formulations and other chemistry using industries are appropriate for the new developing supply chain based on sustainable renewable feedstocks?

Our current approach to formulated product design is based on heuristic knowledge of formulators that allows selecting individual compounds from a library of available materials with known properties. We speculate that most of the compounds (or functions) that make-up the product to be designed can potentially be obtained from a single or very few bio-sources. In this case, it may be possible to design a sequence of transformations required to transform feedstocks into products with desired properties, analogous to a metabolic pathway of a complex organism. In this thesis, we conceptualise some novel approaches to processing bio-feedstocks with the aim of bypassing the step of a fixed library of ingredients. Two approaches are brought forward: one making use of knowledge-based expert systems and the other making use of applications of metabolic engineering and dynamic combinatorial chemistry.

This thesis contributes to the area of chemical product/process design and optimisation. Specifically, it introduces novel techniques for multi-target product/process optimization and categorization of a given product composition according to a particular property. It also introduces novel ideas and concepts for acceleration and automation of processes involved in the design of chemical products.

In summary, the objectives this thesis is addressing are:

1. Development of an approach for sequential, on-line multi-target product/process optimization in a scenario where: automation of the overall design process

is sought; adequate physical models are not available; unknown constraints on the decision space may be present; and resources are limited or costly.

2. Development of an approach that allows to perform categorization of a given product composition according to a particular property, for supervised problems where categorical data are available.

3. Development of new ideas and concepts for acceleration and automation of processes involved in the design of chemical products with predefined properties.

The rest of the thesis is organised as follows. In Chapter 2 we present an approach for sequential on-line multi-target optimization in a scenario where: automation of the overall design process is sought; adequate physical models are not available; unknown constraints on the decision space may be present; and resources are limited or very costly. In chapter 3 we describe application of the approach presented in chapter 2 to on-line multi-target optimization of a copolymerization reaction. In Chapter 4 we introduce an approach that incorporates dimensionality reduction as part of an algorithm for categorization of a given product composition according to a particular property. We compare prediction results of this approach with several well-established classification models. In this chapter we also investigate the application of a variety of non-linear dimensionality techniques to visualisation of formulated product data. In Chapter 5 we provide insight into and attempt to conceptualise new approaches to chemical product development. In particular, we discuss the integration of expert systems, metabolic engineering and dynamic combinatorial chemistry in designing new ways of processing bio-feedstocks in chemical product design. Finally, in Chapter 6 we provide concluding remarks and consider possible extensions of the ideas and approaches presented in this thesis.

Chapter 2

Sequential Multi-Target Optimization of Expensive-to-Evaluate Functions

2.1 Brief summary of the chapter

In this chapter a novel approach to multi-target optimization of expensive-to-evaluate functions is explored that is based on a combined application of Gaussian Processes, Mutual Information and a Genetic Algorithm. The aim of the approach is to find an approximation to the optimal solution (or the Pareto optimal solutions) within a small budget. The approach is shown to compare favourably with a surrogate based on-line evolutionary algorithm on two synthetic problems.

2.2 Introduction

In target optimization one is concerned not with finding the global optimum (unless the target happens to be one), but rather with finding solutions associated with desired values of the underlying process/es. Such optimization problems often arise in product design. In particular, when novel materials and/or methods are involved, it is usually required that certain desired specifications/properties of the product are adhered to. For example, in designing a formulated product such as facial cream (using a constantly updated set of ingredients) it is important that certain product properties are achieved, for example, viscosity and transparency, amongst other

characteristics. In general, this poses a challenging engineering problem in that our capacity to accurately predict properties of formulated consumer products based on composition is quite limited, due to the physical complexity of the system [34]. Our ignorance about the underlying process leads to the need for an often large number of interactions with the real system[1]. For some applications the number of interactions that can be performed is limited due to high cost of the resources involved. In these instances an attractive strategy is to sequentially select experiments that are optimal both in terms of experimental design *and* in terms of identification of suitable solution/s. The strategy is usually implemented via employment of a surrogate model for approximation of values of the response variable in combination with a selective, one-at-a-time, sampling strategy where information from past experiments is used to determine the design of the next one. One of the most prominent algorithms for such sequential optimization is Efficient Global Optimization (EGO) [33]. Since its introduction, the algorithm has been adapted for different types of optimization problems, including target optimization. In [37], for instance, the authors make use of the concepts of *desirability* [32] and *virtual observations* [28] to construct an algorithm capable of identifying and, with each iteration, improving on a cluster of solutions that best associate with target values. Even though the algorithm undoubtedly explores globally throughout the search, it is not designed to actively search for solutions that would allow one to gain the most information about the underlying process (i.e. solutions optimal in terms of experimental design). In this thesis a novel approach to sequential multi-target optimization is proposed that explicitly incorporates maximization of the information gain as one of its objectives.

Let $\boldsymbol{x} \in \mathbb{R}^d$; $\boldsymbol{x} \in \Omega$, were d is the dimension of the problem and Ω the decision space, be the vector of values of the input variables, scalars $y(\boldsymbol{x})$ and y^* the corresponding observation obtained via interacting with the real system and a target respectively, and $X_L = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$ the set of candidate solutions (obtained via discretization of the decision space). Then, for a single-target optimization, the objective is to minimise the sum of regrets[2]

$$\sum_{i=1}^{k} |y(\boldsymbol{x}_i) - y^*|, \tag{2.1}$$

[1] This is termed *high throughput* within formulation industry.

[2] $|\cdot|$ is used to mean absolute value or norm one.

where i is the iteration number, in other words to find

$$\boldsymbol{x}^* = \underset{\boldsymbol{x} \in X_L}{\operatorname{argmin}} |y(\boldsymbol{x}) - y^*| \tag{2.2}$$

in as few iterations as possible. The cumulative regret, as in (2.1), i.e. the loss in reward due to not knowing the points associated with the target beforehand, is a performance metric often used in this context [36]. Note, that the situation where there are multiple solutions to (2.2) may arise. In this thesis we rank such solutions using metric such as the *hypervolume indicator* [38], as explained further in the chapter. Alternatively, this scenario could be handled by the end user, ranking the solutions by applying relevant cost functions. For instance, if the solutions are compositions of ingredients in a formulation, then they can be further ranked in terms of cost of ingredients, or the proportion of a particular ingredient, etc.

The procedure for sequential single-target optimization in an expensive-to-evaluate function scenario could be:

1. Construct a cheap to evaluate surrogate model and train it on a sample of points $X_T = \{(\boldsymbol{x}_1; y(\boldsymbol{x}_1)), (\boldsymbol{x}_2; y(\boldsymbol{x}_2)), ..., (\boldsymbol{x}_h; y(\boldsymbol{x}_h))\}$, $h \ll n$ (the training set).

2. Using the surrogate model's predictions, select $\boldsymbol{x}^*$ that

   - Maximizes information gain about the underlying process - *exploration.*
   - Minimizes predicted regret, i.e. satisfies (2.2) - *exploitation.*

3. Evaluate $\boldsymbol{x}^*$ via interacting with the real system and obtain $y(\boldsymbol{x}^*)$.

4. Include the pair $(\boldsymbol{x}^*, y(\boldsymbol{x}^*))$ in the training set and update the model.

5. Iterate until there is no improvement on the current optimum or the available budget has expired. In the situation where the available budget has expired prior to obtaining a satisfactory solution, the solution/s satisfying (2.2) is chosen from the solutions collected so far.

The above procedure requires that, at each iteration, two objectives are optimized simultaneously. One way to go about it is to search for a set of non-dominated[3] solutions - a Pareto set. A good way to select just one $\boldsymbol{x}^*$ from a pareto set is to choose a solution with the highest value of *hypervolume indicator*. The hypervolume

[3] A solution is non-dominated if it cannot be superseded by another solution which improves an objective without worsening another one.

indicator is the volume of the fitness space that is dominated by a solution[4] and is bounded by a reference point (see Figure 2.1). Also, the procedure entails choosing $\boldsymbol{x}^*$ from a predetermined decision set. Although practical in terms of computational speed, this is limiting in terms of accuracy. It would, hence, be beneficial to find a compromise between the increase in the level of discretization and the search of the entire decision space. The proposed compromise is to first, identify a set of non-dominated solutions from a discrete set of candidate solutions, then, search in the neighbourhood of these solutions, using a real-coded genetic algorithm (non-dominated sorting genetic algorithm II (NSGA-II) [29] is employed), for a solution with the highest value of hypervolume indicator.

Industrial target optimization problems, however, often involve more than one target. For instance, a formulator may need to optimize a formulation for a particular value of viscosity and opacity. The procedure described above can be extended to a multi-target case by constructing a separate surrogate model for each underlying process and for $\boldsymbol{x}^*$, replacing the values of information gain and regret associated with it with chosen aggregates. In this thesis we use the magnitude of a vector whose entries are the values of information gain/regret as the aggregates. After the termination of the multi-target optimization, all of the non-dominated solutions (in relation to the actual targets) could be identified and presented to the end user for further consideration.

For the surrogate model, Gaussian Processes, as it provides a principled way of assessing uncertainty of the model, has successfully been used in many optimization problems and is the choice for the approach discussed here. As has already been mentioned, the sampling criterion is required to account for the need to both exploit the knowledge acquired so far as well as gain more knowledge about the underlying processes. The former can be achieved by choosing $\boldsymbol{x}^*$ that, according to the surrogate model's prediction will take us closer to the target, the latter can be achieved, by choosing $\boldsymbol{x}^*$ that is predicted to reduce the uncertainty about the rest of the input space the most, which can be interpreted as the greatest increase in mutual information [31] between $X_T \cup \boldsymbol{x}^*$ and the rest of the input space. Gaussian Processes allows us, through the use of the mean value (for the computation of regret) and variance (for the computation of information gain) of the predicted distributions for the approximated underlying process values, to efficiently deal with both demands on the sampling criterion.

[4]Usually, though, the hypervolume indicator is computed for a pareto set.

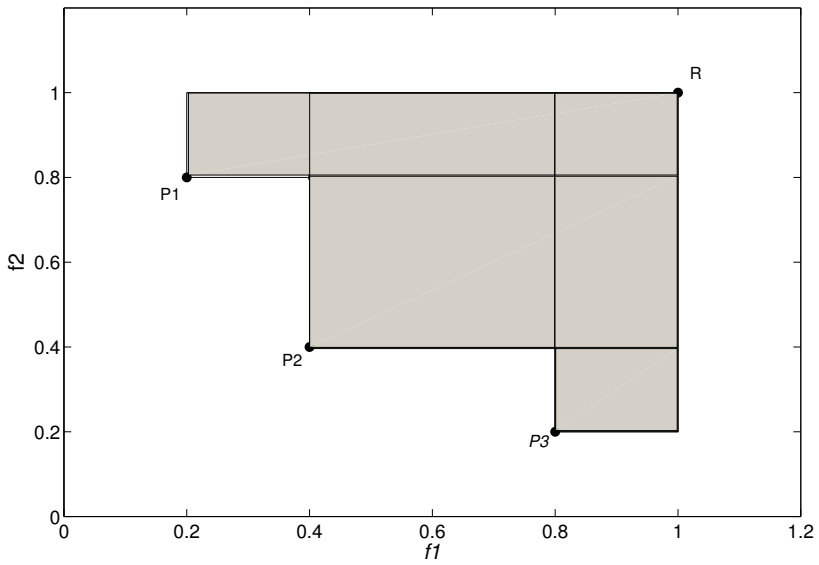


Figure 2.1: The hypervolume (shaded area), bounded by a reference point R, of a three point Pareto set $\{P1, P2, P3\}$. The hypervolume of point $P2$ is the area of rectangle whose diagonally opposite corners are $P2$ and R.

The Efficient Global Optimization (EGO) algorithm would be, because of its successful application to optimisation problems, the best choice for comparison with the approach proposed in this thesis. However, to our knowledge, there is no suitable adaptation of the EGO algorithm for multi-target optimisation. The approach proposed in this thesis is compared with a surrogate based on-line evolutionary algorithm [30] that also uses Gaussian Processes. The algorithm can easily tackle multi-target optimisation problems. It should be noted that other surrogate based on-line evolutionary algorithms exist that are similar in their approach (see, for instance, [40] and [41]). The attractive features of these algorithms are: they have an evolutionary algorithm at the core, capable of solving multi-dimensional, multi-modal problems; they attempt to strike a balance between the need to reduce the amount of expensive evaluations and the need to improve on the quality of the surrogate model. Although both, the approach presented in this work and the algorithm it is compared with, use Gaussian Processes, its application is different, which makes for an interesting comparison. In the rest of the chapter, the proposed algorithm is referred to as a Multi-Objective Active Learner (MOAL) and the abbreviation SOEA is used for the Surrogate based On-line Evolutionary Algorithm.

The chapter is organized as follows. In section 2.3, Gaussian Processes and Mutual Information are, briefly, introduced, followed by a description of the MOAL algorithm in section 2.4. In section 2.5, we describe the application of the MOAL algorithm to two synthetic multi-target optimization problems. In section 2.6 we, briefly, introduce the SOEA algorithm and describe its application to the synthetic multi-target optimization problems. The results are discussed in section 2.7. In section 2.8, we describe an adjustment of the MOAL algorithm for problems that involve unknown constraints on the decision space. Conclusions are drawn in section 2.9.

2.3 Methods

2.3.1 Gaussian Processes Regression

The assumption is that the observations associated with the inputs to the system have a (multivariate) Gaussian joint distribution. Here, a random variable is an observation associated with a particular input. For any subset of the random variables, their joint distribution will also be Gaussian. A Gaussian Process (GP) is a generalization of multivariate Gaussians to an infinite number of random variables. According to a GP definition, the joint distribution over observations associated with every finite subset of inputs is Gaussian. These joint distributions are defined by a GP through the use of mean function, $m(\cdot)$, and covariance function, $k(\cdot,\cdot)$. For an observation associated with input $\boldsymbol{x}$ its mean is given by $m(\boldsymbol{x})$ and for a pair of observations $\boldsymbol{x}$ and $\boldsymbol{x}^{'}$ their covariance, $K(\boldsymbol{x},\boldsymbol{x}^{'})$, is given by $k(\boldsymbol{x},\boldsymbol{x}^{'})$. In this thesis we consider stationary covariance functions. A stationary covariance function is a function of $x-x^{'}$. It is therefore invariant to translations in the input space. A common choice is to apply a Gaussian Process with zero mean function and Automatic Relevance Determination (ARD) Squared Exponential covariance function

$$k(\boldsymbol{x},\boldsymbol{x}^{'}) = \sigma_f^2 \exp\left(-\frac{1}{2}\sum_{d=1}^{D}\left(\frac{x_d - x_d^{'}}{l_d}\right)^2\right) \tag{2.3}$$

In the above, σ_f^2 is the signal variance, l_d is an individual characteristic length scale for each input dimension x_d. However, it is sensible to assume measurement noise to be present. Hence, each observation $y(\boldsymbol{x})$ can be thought of as related to an

underlying process $f(\boldsymbol{x})$ through a Gaussian noise model:

$$y(\boldsymbol{x}) = f(\boldsymbol{x}) + \epsilon, \tag{2.4}$$

were $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$ are independent identically distributed (IID) random errors. And so, the final expression for the covariance function - *the prior on the noisy observations* can be written as

$$k_{final}(\boldsymbol{x}, \boldsymbol{x}^{'}) = \sigma_f^2 \exp\left(-\frac{1}{2}\sum_{d=1}^{D}\left(\frac{x_d - x_d^{'}}{l_d}\right)^2\right) + \delta_{\boldsymbol{x}\boldsymbol{x}'}\sigma_n^2, \tag{2.5}$$

were $\delta_{\boldsymbol{x}\boldsymbol{x}'}$ is the Kronecker delta function. $\boldsymbol{\theta} = \left\{\sigma_f^2, l_1, l_2, ..., l_d, \sigma_n^2\right\}$ forms a set of hyperparameters of the covariance function. To incorporate the knowledge that the training data provide about the process we write the joint distribution of the observed target values and function values at test locations under the prior and then condition it on the observations. The necessary Gaussian identities employed are:

$$\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{bmatrix} A & C \\ C^T & B \end{bmatrix}\right),$$

were $\boldsymbol{x}$ and $\boldsymbol{y}$ are jointly Gaussian random vectors; and the marginal distribution of $\boldsymbol{x}$ together with the conditional distribution of $\boldsymbol{x}$ given $\boldsymbol{y}$

$$\boldsymbol{x} \sim \mathcal{N}\left(\boldsymbol{\mu}_x, A\right),$$

$$\boldsymbol{x}|\boldsymbol{y} \sim \mathcal{N}\left(\boldsymbol{\mu}_x + CB^{-1}\left(\boldsymbol{y} - \boldsymbol{\mu}_y\right), A - CB^{-1}C^T\right).$$

So, now, given a set of inputs to the system, X, corresponding observations, $\boldsymbol{y}$, and an unobserved input $\boldsymbol{x}^*$, for $y(\boldsymbol{x}^*)$

$$\begin{bmatrix} \boldsymbol{y} \\ y(\boldsymbol{x}^*) \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K\left(X, X\right) & K\left(X, \boldsymbol{x}^*\right) \\ K\left(\boldsymbol{x}^*, X\right) & K\left(\boldsymbol{x}^*, \boldsymbol{x}^*\right) \end{bmatrix}\right), \tag{2.6}$$

where each entry in $K(\cdot, \cdot)$ is computed using (2.5). The conditional probability, $P(y(\boldsymbol{x}^*)|\boldsymbol{y})$, then follows a Gaussian distribution:

$$y(\boldsymbol{x}^*)|\boldsymbol{y} \sim \mathcal{N}\left(K\left(\boldsymbol{x}^*, X\right)K\left(X, X\right)^{-1}\boldsymbol{y}, K\left(\boldsymbol{x}^*, \boldsymbol{x}^*\right) - K\left(\boldsymbol{x}^*, X\right)K\left(X, X\right)^{-1}K\left(X, \boldsymbol{x}^*\right)\right).$$

The mean and variance of this distribution are used to compute the best estimate and the uncertainty of $y(\boldsymbol{x}^*)$:

$$\bar{y}(\boldsymbol{x^*}) = K\left(\boldsymbol{x}^*, X\right) K\left(X, X\right)^{-1} \boldsymbol{y}, \tag{2.7}$$

$$\sigma^2_{y(\boldsymbol{x}^*)} = K\left(\boldsymbol{x}^*, \boldsymbol{x}^*\right) - K\left(\boldsymbol{x}^*, X\right) K\left(X, X\right)^{-1} K\left(X, \boldsymbol{x}^*\right). \tag{2.8}$$

A number of models can be constructed depending on the choice of the values of the hyperparameters in the covariance function. Figure 2.2 illustrates two output vectors where the mean and variance of each output have been computed using (2.7) and (2.8), but where the values of the hyperparameters of the covariance function are different. To choose the best model for the data available, a search is carried out for the values of the hyperparameters that maximize the marginal likelihood - the probability of the data given the hyperparameters,

$$\log p(\boldsymbol{y}|X, \boldsymbol{\theta}) = -\frac{1}{2}\boldsymbol{y}^T K^{-1}\boldsymbol{y} - \frac{1}{2}\log|K| - \frac{n}{2}\log 2\pi,$$

where $|K|$ is a determinant of matrix K and n is the number of training examples. As the mean function is set to 0, it's values do not appear in the expression of the marginal likelihood. To set the hyperparameters, partial derivatives of the marginal likelihood w.r.t. the hyperparameters are obtained and used in conjunction with a gradient based optimizer. The advantage of using marginal likelihood is that it automatically incorporates a trade-off between model fit and model complexity. Note, that in order to account for uncertainty in the hyperparameters, integration over the posterior distribution of the hyperparameters is required. The integral over the posterior of the hyperparameters often is not analytically tractable, though. A commonly employed approach is to use numerical integration via Markov chain Monte Carlo (MCMC)-methods. But, due to the computational burden, this approach may be too costly for large data sets. To train a GP model, a choice has to be made (utilizing prior knowledge) between different functional forms for the mean and covariance functions as well as adaptation of the hyperparameters of these functions. In the absence of prior knowledge, a variety of functional forms could be investigated via comparison of the marginal likelihoods (for more details see [35]). In this thesis, only GP with zero mean function and Squared Exponential (ARD) covariance function and GP with zero mean function and Matérn covariance function, as described in Section 2.5 are considered.

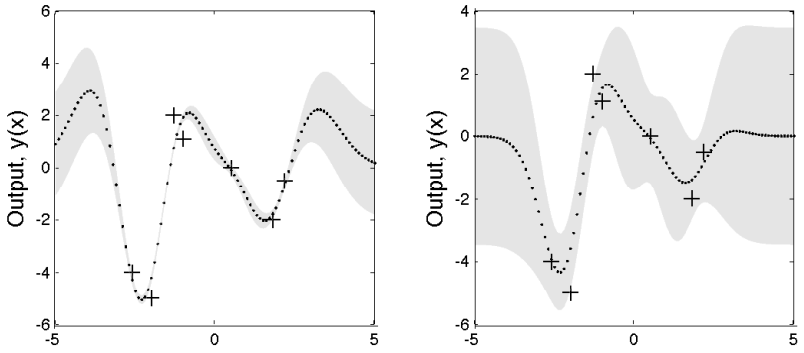


Figure 2.2: Two output vectors where the mean and variance of each output have been computed using (2.7) and (2.8), but where the values of the hyperparameters of the covariance function are different. The training data are shown by + signs. The output predictions (dots) were generated using a GP with the covariance function as in (2.5) with: left - $(l, \sigma_f, \sigma_n) = (1,\ 1,\ 0.1)$; right - $(l, \sigma_f, \sigma_n) = (\sqrt{3}, 0.5, 0.8)$. Both plots also show the 2 standard-deviation error bars for the predictions obtained using these values of the hyperparameters.

2.3.2 Mutual Information

Mutual information of two random variables X and Y measures how much knowing one of these variables reduces uncertainty about the other. It is expressed as

$$MI\left(X;Y\right) = H\left(X\right) - H\left(X \mid Y\right), \tag{2.9}$$

were $H\left(X\right)$ is a marginal entropy - the amount of uncertainty about random variable X and $H\left(X \mid Y\right)$ is a conditional entropy - the amount of uncertainty remaining about X after Y is known, computed as

$$H(X|Y) = H(X,Y) - H(Y). \tag{2.10}$$

The above gives support to the intuition behind the meaning of mutual information as the reduction in uncertainty that knowing one variable provides about the other. Mutual information is non-negative and symmetric. Namely,

$$MI(X;Y) \geq 0 \tag{2.11}$$

and

$$MI(X;Y) = MI(Y;X).$$

For a random variable X with a probability density function f whose support set is $\mathbb{X}$, the differential entropy $H(X)$ is defined as

$$H(X) = -\int_{\mathbb{X}} f(x) \log f(x) d(x). \tag{2.12}$$

The differential entropy of a Gaussian random variable X conditioned on variable Y is a monotonic function of its variance:

$$H(X|Y) = \frac{1}{2} \log \left(2\pi \exp \sigma^2_{X|Y} \right). \tag{2.13}$$

If X is assumed to be an observation associated with a particular input to the system as discussed in the previous section, for instance, then, using GP regression, predicted value of $\sigma^2_{X|Y}$ is easily computed using (2.8). Also, as the computation in (2.8) only depends on the inputs, it is possible to compute $H(X|Y)$ before the actual observation is made. It is useful, in the context of optimization, to think of a discretized input space as a set of random variables. Let X_L be such a set of random variables, X_T be any subset of X_L and x any random variable in $X_L \setminus X_T$, then the mutual information $MI\left(X_L \setminus X_T \cup \boldsymbol{x}; X_T \cup \boldsymbol{x}\right)$, expressed as

$$MI\left(X_L \setminus X_T \cup \boldsymbol{x}; X_T \cup \boldsymbol{x}\right) = H\left(X_L \setminus X_T \cup \boldsymbol{x}\right) - H\left(X_L \setminus X_T \cup \boldsymbol{x} \mid X_T \cup \boldsymbol{x}\right),$$

is the information gain (or the amount of uncertainty remaining about $X_L \setminus X_T \cup \boldsymbol{x}$ (the rest of the input space)), if $\boldsymbol{x}$ is revealed.

2.4 MOAL algorithm

We assume the underlying physical processes are unknown. The experimental data (real observations) are used to train the surrogate models that approximate the underlying unknown physical processes.

Consider n objectives, associated with n targets, as in (2.2), and a budget of t evaluations. Two sets of points are involved: the training set, X_T, which is used to train the surrogate models and which gains a point with each iteration of the algorithm; and a set, X_L, of other solutions from a discretised decision space. A surrogate model is trained for each process. Then at each iteration of the algorithm:

1. Using the surrogate models, estimates and the associated predicted variances of observations for each point $\boldsymbol{x}_j \in X_L$, $j = 1, ..., |X_L|$, are computed.

2. In relation to i's objective, every point is referenced in two ways:

   (a) The increase in mutual information it would provide [31]:

$$\begin{aligned}
\mathcal{I}^{(i)}(\boldsymbol{x}_j) &= MI(X_T \cup \boldsymbol{x}_j; X_L \setminus X_T \cup \boldsymbol{x}_j) - MI(X_T; X_L \setminus X_T) \qquad (2.14)\\
&= H(X_T \cup \boldsymbol{x}_j) - H(X_T \cup \boldsymbol{x}_j \mid X_L \setminus X_T \cup \boldsymbol{x}_j)\\
&\quad - [H(X_T) - H(X_T \mid X_L \setminus X_T)]\\
&= H(X_T \cup \boldsymbol{x}_j) - H(X_T)\\
&\quad -H(X_T \cup \boldsymbol{x}_j \mid X_L \setminus X_T \cup \boldsymbol{x}_j) + H(X_T \mid X_L \setminus X_T)\\
&= H(\boldsymbol{x}_j \mid X_T) - [H(X_L) - H(X_L \setminus X_T \cup \boldsymbol{x}_j)]\\
&\quad +H(X_L) - H(X_L \setminus X_T)\\
&= H(\boldsymbol{x}_j \mid X_T) - [H(X_L \setminus X_T) - H(X_L \setminus X_T \cup \boldsymbol{x}_j)]\\
&= H(\boldsymbol{x}_j \mid X_T) - H(\boldsymbol{x}_j \mid X_L \setminus X_T \cup \boldsymbol{x}_j)\\
&= \frac{1}{2}\log 2\pi e \sigma^{2^{(i)}}_{(\boldsymbol{x}_j \mid X_T)} - \frac{1}{2}\log 2\pi e \sigma^{2^{(i)}}_{(\boldsymbol{x}_j \mid X_L \setminus X_T \cup \boldsymbol{x}_j)}\\
&= \frac{1}{2}\log\left(\frac{\sigma^{2^{(i)}}_{(\boldsymbol{x}_j \mid X_T)}}{\sigma^{2^{(i)}}_{(\boldsymbol{x}_j \mid X_L \setminus X_T \cup \boldsymbol{x}_j)}}\right)
\end{aligned}$$

   where (2.9), (2.10) and (2.13) are employed to write $H(X_T \cup \boldsymbol{x}_j) - H(X_T)$ as $H(\boldsymbol{x}_j \mid X_T)$, $H(X_T \cup \boldsymbol{x}_j \mid X_L \setminus X_T \cup \boldsymbol{x}_j)$ as $H(X_L) - H(X_L \setminus X_T \cup \boldsymbol{x}_j)$, $H(X_T \mid X_L \setminus X_T)$ as $H(X_L) - H(X_L \setminus X_T)$ and equation (2.8) is used to compute $\sigma^{2^{(i)}}_{(\boldsymbol{x}_j \mid X_T)}$ and $\sigma^{2^{(i)}}_{(\boldsymbol{x}_j \mid X_L \setminus X_T \cup \boldsymbol{x}_j)}$.

   (b) The predicted value of regret

$$r^{(i)}(\boldsymbol{x}_j) = |\bar{y}^{(i)}(\boldsymbol{x}_j) - y^{*(i)}|, \qquad (2.15)$$

   where $\bar{y}^{(i)}(\boldsymbol{x}_j)$ and $y^{*(i)}$ are the predicted value of the response variable i at $\boldsymbol{x}_j$ and the target value of the response variable i respectively.

3. The sets $\left\{\mathcal{I}^{(i)}(\boldsymbol{x}_1), \mathcal{I}^{(i)}(\boldsymbol{x}_2), ..., \mathcal{I}^{(i)}(\boldsymbol{x}_{|X_L|})\right\}$ and $\left\{r^{(i)}(\boldsymbol{x}_1), r^{(i)}(\boldsymbol{x}_2), ..., r^{(i)}(\boldsymbol{x}_{|X_L|})\right\}$ are mapped onto the interval $[0, 1]$. The information about every point is, first,

summarized as

$$\boldsymbol{\mathcal{I}}(\boldsymbol{x}_j) = \begin{bmatrix} \mathcal{I}^{(1)}(\boldsymbol{x}_j) \\ \mathcal{I}^{(2)}(\boldsymbol{x}_j) \\ \cdots \\ \mathcal{I}^{(n)}(\boldsymbol{x}_j) \end{bmatrix}, \quad \boldsymbol{r}(\boldsymbol{x}_j) = \begin{bmatrix} r^{(1)}(\boldsymbol{x}_j) \\ r^{(2)}(\boldsymbol{x}_j) \\ \cdots \\ r^{(n)}(\boldsymbol{x}_j) \end{bmatrix}, \tag{2.16}$$

then, the magnitudes $\|\boldsymbol{\mathcal{I}}(\boldsymbol{x}_j)\|_2$ and $\|\boldsymbol{r}(\boldsymbol{x}_j)\|_2$ are computed. The following procedure[5] is then used to choose one point for sampling:

(a) All of the points are sorted according to non-domination, using the magnitudes $\|\boldsymbol{\mathcal{I}}(\boldsymbol{x}_j)\|_2$ and $\|\boldsymbol{r}(\boldsymbol{x}_j)\|_2$, and a Pareto set,$\chi$, identified. The point, $\boldsymbol{x}_c$, satisfying

$$\boldsymbol{x}_c = \underset{\boldsymbol{x}_j}{\operatorname{argmax}} \|\boldsymbol{\mathcal{I}}(\boldsymbol{x}_j)\|_2 \times \left(\sqrt{n} - \|\boldsymbol{r}(\boldsymbol{x}_j)\|_2\right) \tag{2.17}$$

is chosen as the 'current best'. Set χ is, first, reduced to size $z \ll |X_L|$ (to include only $\boldsymbol{x}_c$ and at most $z-1$ 'next best' non-dominated points selected according to (2.17)) and, then, used as the first population for a NSGA-II algorithm (without crossover) to conduct a search for the maximizer.

(b) The NSGA-II algorithm is iterated M times. At each iteration:

- Following sorting and selection steps, mutations are carried out (by performing small perturbations of the input vectors) to obtain a set (of size $|\chi|$) of new points within the decision space.
- Each of the new points is referenced using (2.14) and (2.15) and one point is chosen according to (2.17). If the value computed for it using (2.17) is higher than that one of the 'current best' point, it becomes the 'current best' point.
- The 'current best' point after the Mth iteration is chosen for evaluation.

4. The evaluated point is added to the training set and the hyperparameters of the surrogate models are reoptimized.

Note that for points in X_L that are near the one that was just sampled their predicted variance, $\sigma^2_{(\boldsymbol{x}_j|X_T)}$ (see last line in equation (2.14)) will decrease, as will the

[5]Note, that other, alternative to Genetic algorithm, techniques (branch-and-bound algorithm, for instance) could be used to find a solution to the single objective, constrained (by the boundaries of the decision space) optimization problem posed in (2.17).

associated value of the increase in mutual information. This means that, potentially, the points close to the target will not be sampled for a number of iterations. To overcome this problem, the decision set X_L is re-sampled with density $\hat{p}_{X_L^*|\tilde{\boldsymbol{x}}}$ (where $\tilde{\boldsymbol{x}}$ is a matrix of solutions collected so far), when there has been no improvement in the value of the hypervolume indicator of the Pareto set for a number[6] of consecutive iterations. Re-sampling with $\hat{p}_{X_L^*|\tilde{\boldsymbol{x}}}$ is done using a mixture of Gaussians (with n components) as a density estimator. The value of the hypervolume indicator of the Pareto set is obtained as follows: first, the observations $\left\{y_1^{(1)}, ..., y_1^{(n)}; y_2^{(1)}, ..., y_2^{(n)}; ...; y_m^{(1)}, ..., y_m^{(n)}\right\}$, collected so far, are transformed as

$$\frac{|y_j^{(i)} - y^{*(i)}|}{R^{(i)}}, i = 1, ..., n \;\; j = 1, ..., m \tag{2.18}$$

where

$$R^{(i)} = \max |y_j^{(i)} - y^{*(i)}|; \tag{2.19}$$

then, from the transformed observations, the non-dominated set is identified and the value of the hypervolume indicator (for the whole of the non-dominated set), bounded by a reference point, which is a vector of ones and of length m, computed.

As discussed above, relevant to the accuracy of the algorithm, the signal for the resampling of the decision set could be when there has been no improvement in the value of the hypervolume indicator of the Pareto set for a number of iterations. The same could be applied as a stopping criteria for the overall algorithm. However, there is danger of stopping too early. Instead (or additionally) the algorithm could be stopped ones a solution is found that is within an acceptable error away from the target. However, there is no guarantee that solutions within the predefined error exist, in which case there is danger of performing more experiments than necessary. In this thesis, for the lack of a robust error based stopping rule, we stop the algorithm once the budget of evaluations is exhausted. The final Pareto set is presented to the end user. For comparison of performance against other algorithms (or against optimum performance, if such information is available), the value of the hypervolume indicator for the final Pareto set (bounded by a reference point, which is a vector of ones and of length m) can also be computed. To reduce computational complexity, $\sigma^{2(i)}_{(\boldsymbol{x}_j|X_L\setminus\boldsymbol{x}_j)}$ is calculated using only k points, where

$$2|X_T| \leq k \leq |X_L \setminus \boldsymbol{x}_j|. \tag{2.20}$$

[6]The number is linked to the dimensionality of the problem. For a $2d$ problem we wait for 4 iterations, whereas for a $6d$ problem we would wait for at least 10 iterations.

Namely, points $\boldsymbol{x}^{'} \in X_L \setminus \boldsymbol{x}_j$ are arranged in decreasing order according to their respective values of covariance with $\boldsymbol{x}_j$ (computed using (2.5)) and the first k are selected.

It should be noted that, although presented specifically as a multi-target optimization algorithm, MOAL can still be applied in situations where some objectives are global. In such cases all that is required is a suitable 'unreachable' target. For instance, if minimization of monetary cost is one of the objectives, it can be converted into a target of £0.00. In chapter 3 an optimization problem with one target and one global objective is discussed.

2.5 Illustration of the approach

To illustrate the potential use of the approach, it is applied to simulate two multi-target optimization problems. The Ackley [42], the Booth [43], the Levy [44] and the Dixon & Price [45] functions where employed to simulate fictitious physical processes. These functions are widely used for testing optimization algorithms. The first problem illustrates the application of the algorithm to a two-target unconstrained optimization problem, where: the two fictitious physical processes are simulated by the Ackley and the Booth functions (see Figure 2.3)

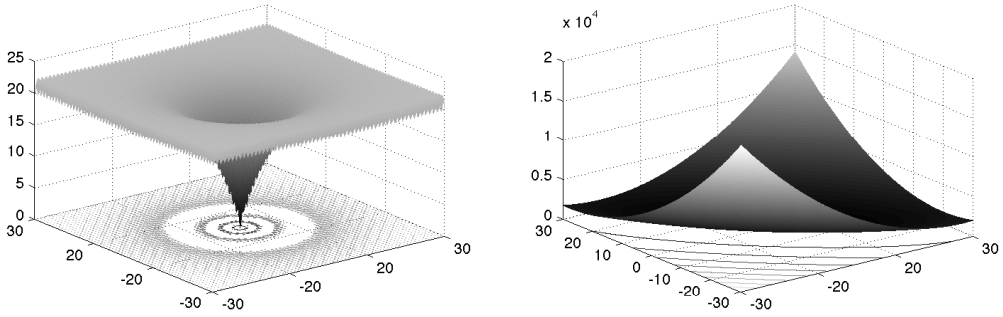


Figure 2.3: Left: the Ackley and right: the Booth functions in two dimensions.

$$f_{\text{Ackley}}(x,y) = - 20\exp\left(-0.2\sqrt{0.5\left(x^2+y^2\right)}\right) - \exp\left(0.5\left(\cos\left(2\pi x\right)+\cos\left(2\pi y\right)\right)\right) + 20 + \exp\left(1\right), \tag{2.21}$$

$$\text{where} \quad f_{\min}\left(0,0\right) = 0$$

$$f_{\text{Booth}}(x, y) = (x + 2y - 7)^2 + (2x + y - 5)^2 , \tag{2.22}$$

$$\text{where} \quad f_{\min}(1, 3) = 0;$$

the number of input variables is two, with each one ranging from -30 to 30; and both targets are global minimums. Thus, the target vector is $[0\ 0]^T$. The set of candidate solutions, X_L, comprises of 1200 input vectors uniformly spread out over the decision space. The initial training set, X_T, comprises of 16 input vectors, obtained as a Latin Hyper Cube (LHC) [46] sample, and corresponding values of two processes. The values of the Booth function are log transformed prior to regression. For this problem a GP with zero mean and the Matérn covariance function was employed

$$k_{\text{Matérn}}\left(\boldsymbol{x}, \boldsymbol{x}^{'}\right) = \sigma_f^2 \frac{2}{1-\nu} \Gamma(\nu) \left(\frac{\sqrt{2\nu} r}{l}\right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu} r}{l}\right), \tag{2.23}$$

with positive parameters ν, σ_f^2 and l, where K_ν is a modified Bessel function and $r = |\boldsymbol{x} - \boldsymbol{x}^{'}|$. $\nu = \frac{3}{2}$ was chosen for this problem, for which (2.23) can be simplified [27] to

$$k_{\text{Matérn}}\left(\boldsymbol{x}, \boldsymbol{x}^{'}\right) = \sigma_f^2 \left(1 + \frac{\sqrt{3} r}{l}\right) \exp\left(-\frac{\sqrt{3} r}{l}\right). \tag{2.24}$$

The parameters σ_f^2 and l in (2.23) and (2.24) play the same role as in (2.3). Note, there is just one length parameter, as opposed to one per dimension of the problem.

The first problem is challenging as, in order to approximate the optimal Pareto set well, the algorithm is required to find solutions that are near both global minima. A big proportion of the landscape of the Ackley function is featureless, thus, a surrogate model trained on a small initial training set may not be able to produce satisfactory predictions for points in the target area, and the algorithm will need to explore efficiently (i.e. update the surrogate model with most informative points quickly), for an optimization to converge on a satisfactory set of solutions within a small budget of evaluations. For the Booth function, the global optimum is inside a long, flat valley. To find the valley is not difficult, however, convergence to the global optimum is challenging. For the Ackley function, the global optimum is inside a narrow funnel, making it also non-trivial to locate.

The second problem illustrates the application of the algorithm to a two-target constrained optimization problem, where: the two fictitious physical processes are

simulated by the Levy and the Dixon & Price functions

$$f_{\text{Levy}}(\boldsymbol{x}) = \sin^2(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 \left[1 + 10\sin^2(\pi y_{i+1})\right] + (y_n - 1)^2, \quad (2.25)$$

were

$$y_i = 1 + \frac{x_i - 1}{4},$$

$$f_{\text{DixonPrice}}(\boldsymbol{x}) = (x_i - 1)^2 + \sum_{i=2}^{n} i \left(2x_i^2 - x_{i-1}\right)^2; \quad (2.26)$$

To mimic problems encountered in formulation industry where the problems are often moderate dimensional, we set the number of input variables to four. A constraint often encountered in industrial applications is also applied

$$\sum_{i=1}^{4} x_i = T, \quad (2.27)$$

were $x_i \in \mathbb{R}_{\geq 0}$ and T is user defined ($T = 10$ is used for this problem). The situation is often encountered in experiments with formulated products, for instance, where x_i are volumes of ingredients and T is the total volume per formulation. Three random target vectors were chosen from the $\left[f_{min}^{(1)}(\boldsymbol{x}), f_{max}^{(1)}(\boldsymbol{x})\right] \times \left[f_{min}^{(2)}(\boldsymbol{x}), f_{max}^{(2)}(\boldsymbol{x})\right]$ box. The set of feasible solutions, X_L, comprises of 2000 uniformly spread out input vectors satisfying (2.27). The initial training set, X_T, comprises of 30 uniformly spread out feasible input vectors and corresponding values of two processes. For this problem a GP with zero mean and Squared Exponential (ARD) covariance function was employed.

For both problems:

- Process values are perturbed by noise drawn from $\mathcal{N}(0, 0.1^2)$
- In (2.20), $k = 2|X_T|$ is used
- The NSGA-II algorithm is iterated 100 times per iteration of the main algorithm with a population size at most 1% of $|X_L|$. For each mutation, the value of the perturbation is drawn from the uniform distribution on the interval $(0, \alpha 60]$ and $(0, \alpha 10]$ for the first and second problem respectively. Values of the parameter α from the interval $[0.01, 0.1]$ were tested and $\alpha = 0.01$ selected.
- The decision set is re-sampled, if there has been no improvement in the value of the hypervolume indicator for three consecutive iterations. To re-sample,

only the solutions so far collected, $X_{\boldsymbol{x}^*}$, are considered. $X_{\boldsymbol{x}^*}$ are assumed to belong to a mixture of Gaussian distributions with the number of components that of the dimension of the input space. A Gaussian mixture model is fitted [26] and the parameter estimates (components' means, covariances and mixture proportions) are obtained using an Expectation Maximization (EM) algorithm. A set of random input vectors X_L^* (of the same cardinality as X_L) is then drawn from the resulting distribution.

The hyperparameters[7] of surrogate models are fitted by optimizing the marginal likelihood using a conjugate gradient optimizer. To avoid bad local minima 5 random restarts are tried, picking the run with the best marginal likelihood. Leave-one-out cross-validation is used to validate the models. Namely, for each point in the training set, its predicted function value along with the variance of the predicted value are computed using the rest of the set. Following [33], cross-validated standardized residuals, S_r, are computed

$$S_{r_{\boldsymbol{x}}} = \frac{y(\boldsymbol{x}) - \bar{y}(\boldsymbol{x})}{\sqrt{\sigma^2_{y(\boldsymbol{x})}}}, \tag{2.28}$$

and a check is carried out that the standardized residuals are all in the interval $[-3, +3]$. When training surrogate models in problem 1, some of the computed standardized residuals failed the test. In an attempt to improve the fit of the GPR models we log transformed the values of both the Ackley and the Booth functions. The log transformation worked well and the standardized residuals where all inside the interval $[-3, +3]$. In general, if the GPR model fails the validation test, we would try two transformations: the log transformation, $\log(y)$, and the inverse transformation, $-1/y$. If this does not help, we would investigate the possibility of using a non-stationary covariance function. The optimizations are run for 30 iterations. The observations thus collected are transformed using (2.18). From the transformed observations, the non-dominated set is identified and the value of the hypervolume indicator for the whole of the set (bounded by a reference point $[1, 1]$) computed. The value is then compared against the one computed for the SOEA algorithm and the optimum or a suitably chosen baseline. In this thesis, a baseline is obtained by computing the value of the hypervolume indicator for non-dominated observations obtained having evaluated 10000 uniformly spread out input vectors.

[7] Gaussian Process Regression and Classification Toolbox version 3.1 for Matlab by Carl Edward Rasmussen and Hannes Nickisch downloaded from http://gaussianprocess.org/gpml/code was used.

2.6 Brief description of the SOEA algorithm for multi-target optimization

In this work the approach proposed by [30] is adapted. The attractive features of this approach are: it has an evolutionary algorithm at the core, capable of solving multi-dimensional multi-modal optimisation problems; it attempt to strike a balance between the need to reduce the amount of expensive evaluations and the need to improve on the quality of the surrogate model. Essentially, each iteration of the approach consists of two steps: the search step and the function evaluation step. The search step involves the use of selection, crossover and mutation operators to create a new population of solutions, promising in terms of proximity to the target. The function evaluation step involves the use of a GPR model for approximation of the function values associated with each new population of solutions and identification of solutions (within the new population) that need to be evaluated via real experiments.

The SOEA algorithm proceeds as follows:

1. The initial population of solutions of size N is chosen. The initial population of solutions and the corresponding observations are used as a training set to train surrogate models.

2. A GP with zero mean and Squared Exponential (ARD) covariance function is used. The surrogate models' set up, validation and hyperparameter optimization follows that described in section 2.5.

3. Using a multi-objective evolutionary algorithm (NSGA-II), the next population of solutions is obtained.

4. The trained surrogate models are used to predict the mean values and the corresponding variances of the process values (see equations (2.7) and (2.8))) for each of the obtained solutions. From the predicted variances of the process values, corresponding standard deviations are computed and normalized to be in the interval $[0, 1]$.

5. Solutions for which the normalized standard deviation of each predicted process value is below the currently allowable tolerance, Tolerance_c, are assigned the predicted process values. For the rest, the values are established through interaction with the real system. These points are added to the training set.

The value of Tolerance_c is updated after each iteration of the overall algorithm. It is reduced as follows:

$$\text{Tolerance}_c^{(i)} = \text{Tolerance}_m \times \frac{t - \text{Total}_s^{(i-1)}}{t - N}, \tag{2.29}$$

where Tolerance_m is the maximum allowable tolerance (initialized prior to optimization), t is the maximum number of interactions with the real system that are budgeted for, $\text{Total}_s^{(i-1)}$ is the total number of solutions (up to the iteration $i-1$) that were evaluated via interacting with the real system, and N is the number of solutions in a population. Prior to the first iteration, Tolerance_c is equal to Tolerance_m. To avoid an infinite loop scenario, where evaluations are carried out using the surrogate models only, Tolerance_m is reduced by half if, at i^{th} iteration all of the solutions in the population have been assigned their predicted values.

6. The hyperparameters of the surrogate models are reoptimized after each iteration. The overall algorithm is iterated until the budget is exhausted.

Once the budget of evaluations has been exhausted, the algorithm is stopped. The corresponding observations are transformed using (2.18). From the transformed observations, the non-dominated set is identified and the value of the hypervolume (using ($[1, 1]$ as a reference point) computed. The same decision set and the initial training set as for the MOAL algorithm where used. The initial value of Tolerance_m (i.e. the initial maximum allowable normalised standard deviation for the predicted process values) parameter was established through experimentation. Values between 0.05 and 0.5 were tested and a value of 0.1 selected.

2.7 Results and Discussion

The MOAL and SOEA algorithms were tested on the problems presented in section 2.5. Ten optimization runs were performed for each target vector and the mean values of the hypervolume indicator of the Pareto set, along with corresponding standard deviations, were recorded (see Tables 2.1 and 2.2). These values were used to compare the performance of the algorithms. For both algorithms the $R^{(i)}$ in (2.18) were computed[8] using observations from 10000 uniformly spread out solutions.
As can be seen from the results, the MOAL algorithm has performed better than the SOEA on both problems. The plausible explanation is that the MOAL algorithm is

[8]In real application these values would be established using all available observations after the last iteration of the algorithm.

MOAL	SOEA	Baseline
49.43%(4.11)	27.55%(15.86)	64.66%

Table 2.1: Mean and standard deviation of the hypervolume indicator of the Pareto set for the target vector in problem 1 (after 10 runs of the algorithms). The Baseline column refers to the value of the hypervolume indicator of the non-dominated observations obtained by evaluating 10000 uniformly spread out input vectors.

	MOAL	SOEA	Optimum/Baseline
Target vector 1	98.44%(0.68)	91.71%(3.90)	100%
Target vector 2	87.01%(3.31)	81.98%(4.00)	93.12%
Target vector 3	97.92%(0.50)	91.84%(1.78)	100%

Table 2.2: Mean and standard deviation of the hypervolume indicator of the Pareto set for the target vectors in problem 2 (after 10 runs of the algorithms). The Baseline column refers to the values of the hypervolume indicator of the non-dominated observations obtained by evaluating 10000 uniformly spread out input vectors.

able to *actively* improve on the prediction quality of the surrogate models over the target area, and do so rapidly (see Figure 2.4). Locating the areas of the decision space least well covered by the training set, whilst at the same time 'promising' in terms of gaining on the targets, allows the MOAL algorithm to efficiently discover relevant (for the optimization) features of the underlying function landscape. By contrast, the SOEA algorithm is only concerned with reducing uncertainty in the search areas. In a situation where the underlying function landscape is challenging and the budget of the evaluations is small the algorithm can be very successful or unsuccessful depending on how quickly the evolutionary part of it can converge on solutions near the target area/s. This is reflected in the high value of the standard deviation of the hypervolume indicator for problem 1 (see table 2.1).
The following performance criteria can be used to assess the quality of predictions of the surrogate models:

1. Standardized[9] mean squared error (SMSE) loss, which is the mean squared error (MSE) normalized by the variance of the targets of the test cases.

2. The negative log probability (NLP) of the target under the model (since we produce a predictive distribution at each test input),

$$-\log p\left(y_*|X_T, \boldsymbol{x}_*\right) = \frac{1}{2}\log\left(2\pi\sigma_*^2\right) + \frac{\left(y_* - \bar{y}\left(\boldsymbol{x}_*\right)\right)^2}{2\sigma_*^2}, \tag{2.30}$$

[9]MSE, on it's own, is sensitive to the overall scale of the target values.

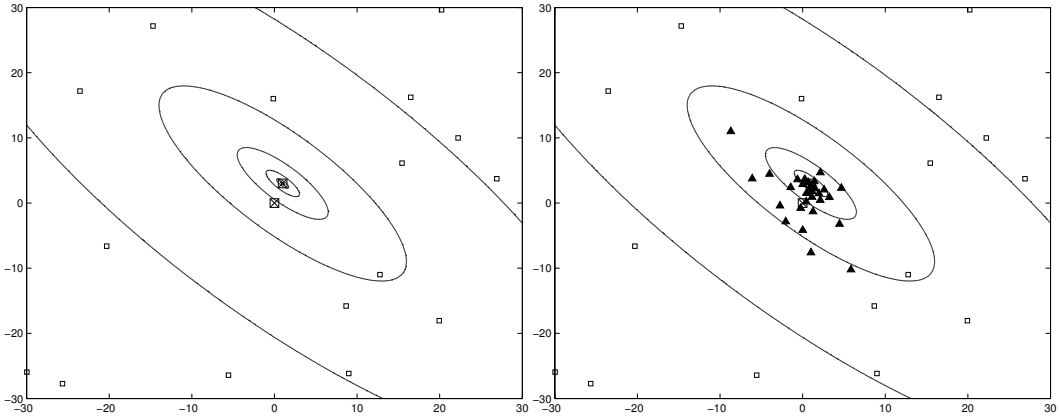


Figure 2.4: Contour plot of the Booth function with, left: input locations of the initial training set and, right: solutions obtained using the MOAL algorithm for an optimization run of 30 evaluations. Empty squares - solutions from the initial training set; filled triangles - solutions chosen by the MOAL algorithm; squares with a cross inside - the global minima of the Booth and the Ackley functions.

where $\bar{y}(\boldsymbol{x}_*)$ and σ_*^2 are the estimated mean and variance of the predictive distribution respectively. This can be summarized by the mean negative log probability (MNLP), by averaging over the test set. This loss can be standardized by computing it relative to the NLP of a predictive model that ignores the inputs and always predicts using a Gaussian distribution with the mean and variance of the training data. The MNLP will, then, be approximately zero for a simple predictive model and negative for a better one. Prediction quality of the surrogate models approximating the Booth function (as in problem 1) is used as an example (see Figures 2.5 and 2.6). The test set is chosen to be the solutions in and around the target area in the ($[-5, 5] \times [-5, 5]$ box).

As can be seen from Figure 2.5, the surrogate models employed to approximate the Booth function during optimization runs of the MOAL algorithm, produced predictions with smaller errors, on average. There is a big drop in the value of SMSE after just 5 evaluations, and steady decrease thereafter, which indicates that the target area was found quickly, and that solutions are being chosen from it (the target area). The MNLP value (see Figure 2.6) also decreases rapidly by 5 evaluations, although the improvement is less pronounced thereafter. It can be argued that for the SOEA algorithm, on average, 30 evaluations were not enough to narrow down the search and, hence, adequately update its surrogate models.

The overall complexity of the MOAL algorithm is mostly due to computation of

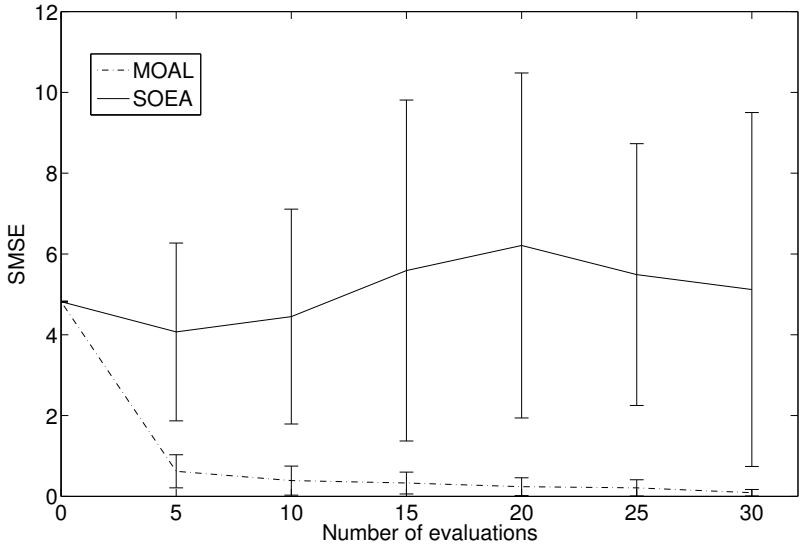


Figure 2.5: Average SMSE values computed for surrogate models approximating the Booth function in problem 1. The averages were computed over 10 optimization runs for budget sizes from 5 to 30 evaluations in increments of 5. Zero evaluations corresponds to the values computed for the model constructed using the initial training set.

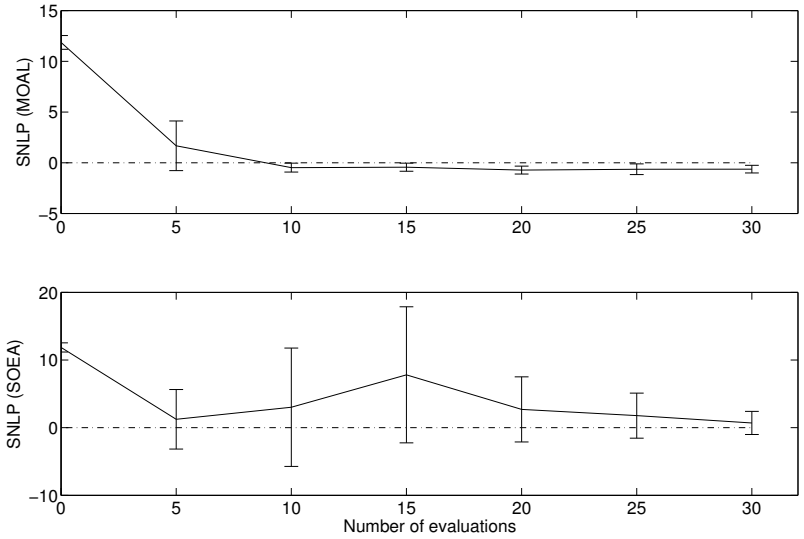


Figure 2.6: Average MNLP values computed for surrogate models approximating the Booth function in problem 1. The averages were computed over 10 optimization runs for budget sizes from 5 to 30 evaluations in increments of 5. Zero evaluations corresponds to the values computed for the model constructed using the initial training set.

the inverse of the covariance matrix (using Cholesky decomposition) for obtaining conditional entropies $H(\boldsymbol{x}_j \mid X_T)$ and $H(\boldsymbol{x}_j \mid X_L \setminus X_T \cup \boldsymbol{x}_j)$ using (2.13) in section 2.3. The computational complexities for the approximation of the conditional entropies are $O\left(t^4|\tilde{X_L}|\right)$ and $O\left(tk^3|\tilde{X_L}|\right)$ for $H(\boldsymbol{x}_j \mid X_T)$ and $H(\boldsymbol{x}_j \mid X_L \setminus X_T \cup \boldsymbol{x}_j)$ respectively, where $|\tilde{X_L}| = |X_L| + M \times z$ (the number of points in the decision space, as per discretization, plus the additional points obtained through the application of the genetic algorithm) and t is the budget (number of evaluations). The increase in complexity comes with increase in the number of evaluations and increase in discretization of the decision space, where the latter is dependant on the dimensionality of the problem. With this in mind, it is thought that the MOAL algorithm is most suited for the problems where the cost of the resources outweighs the computational burden. For instance, a formulator may need to optimize a formulation and have a very limited number of experiments to conduct, due to a high cost of a particular ingredient. Or, in chemical reaction optimization, some processes may require a long time to run their course.

2.8 Algorithm for target optimization where some of the constraints are unknown

It is sometimes the case, in formulated product design, that particular compositions of ingredients lead to the creation of unstable mixtures or, in chemical process design, some input settings may be unworkable. For instance, in emulsion polymerisation reactions, concentration of initiator fed into the system influences the rate of conversion and the size of the polymer particles formed. However, a concentration that is too high may cause a blockage in the tubing part of the reactor. An experienced formulator/chemical engineer may have some knowledge of what compositions/input settings are likely to create unstable mixtures/equipment blockage and/or may even provide insight as to a set of suitable corresponding constraints. Often, though, the necessary constraints are established by conducting a separate, potentially large, set of experiments. In a 'small budget' target optimization scenario, however, this may be wasteful as learning the decision boundary (stable/unstable solutions) for the whole of the decision space may not be necessary for our primary task, i.e. finding the optimal solution/s in relation to the given target. In this work, in order to learn the boundaries of valid regions near the target, we incorporate a classification model into the MOAL algorithm described in section 2.4.

Process optimization, where the constraints are unknown, is not trivial and has re-

ceived a fair amount of attention (see [25], and the references therein). For instance, in [21] the authors investigate the hydraulic capture problem (from the community problems [22]), and present an approach based on treed[10] Gaussian processes [23] for response surface prediction together with a random forests [24] classification algorithm to learn the boundaries of valid regions.

2.8.1 Incorporation of a classification model into the MOAL algorithm

We label stable and unstable solutions as feasible and infeasible respectively. In this work we apply Gaussian Process Classification (GPC) [35] as it allows for robust modelling of class probabilities and a principled approach to evaluating uncertainty of predictions. In more detail, the adjusted MOAL algorithm is:

1. Using only known constraints, create a set of candidate solutions, X_L. Also, choose a small set of candidate solutions, uniformly spread out within the decision space, to form a training set, X_T.

2. Use the feasible solutions from the training set to train a GP regression model for each objective and the whole of the training set to train a two-class [feasible/infeasible] GP classification model for prediction of feasibility.

3. Apply the classification model to predict feasibility of each candidate solution in X_L. The candidate solutions predicted as feasible are selected to form the set $X_L^{'}$.

4. Using $X_L^{'}$ as the set of candidate solutions, perform one iteration of the MOAL algorithm to choose one candidate solution to be evaluated. Evaluate the solution and update the training set. If the evaluated solution is infeasible, the training set is updated with the solution and the label only, otherwise it is updated with the solution, label and the corresponding values of the observations.

5. Go to step 2.

6. Stop once the allowed number of evaluations is exhausted.

[10] Meaning that the input space is partitioned into regions and a separate stationary GP model is fitted within each region.

2.8.2 Gaussian Process (binary) Classification

Let $X = [\boldsymbol{x}_1, ..., \boldsymbol{x}_n]$ be the matrix of inputs and $Z = [z_1, ..., z_n]$ be the vector of corresponding labels, were $z_i \in \{-1, +1\}$ is the class label of input $\boldsymbol{x}_i$. As the output is now discrete, it can not be related to the underlying function values through a noise process, as described in section 2.3.1. Instead, GPC models $p(z|\boldsymbol{x})$ as a Bernouilli distribution, where the probability of success is related to, via a sigmoid function[11], a latent function, $f(\boldsymbol{x})$, i.e.,

$$\pi(\boldsymbol{x}) \triangleq p(z = +1|\boldsymbol{x}) = \sigma(f(\boldsymbol{x})). \tag{2.31}$$

Let $\mathbf{f} = [f_1, ..., f_n]$ be the values of the latent function. For a test case $\boldsymbol{x}^*$, the inference is performed by first: computing the distribution of the latent variable corresponding to $\boldsymbol{x}^*$

$$p(f^*|X, Z, \boldsymbol{x}^*) = \int p(f^*|X, \boldsymbol{x}^*, \mathbf{f}) p(\mathbf{f}|X, Z) d\mathbf{f}, \tag{2.32}$$

where

$$p(\mathbf{f}|X, Z) = \frac{p(Z|\mathbf{f}) p(\mathbf{f}|X)}{p(Z|X)} \tag{2.33}$$

is the posterior over the latent variables, then, using the result in (2.32), to produce a probabilistic prediction

$$\bar{\pi}^* \triangleq p(z^* = +1|X, Z, \boldsymbol{x}^*) = \int \sigma(f^*) p(f^*|X, Z, \boldsymbol{x}^*) df^*. \tag{2.34}$$

The non-Gaussian likelihood in (2.32) makes the integral analytically intractable. This can be overcome by either using an analytic approximation or Monte Carlo sampling. In this work, we place a zero-mean and Automatic Relevance Determination (ARD) Squared Exponential covariance function Gaussian Process prior over the latent function, and employ the expectation propagation (EP) method [39] to analytically approximate the non-Gaussian joint posterior,

$$p(\mathbf{f}|X, Z), \tag{2.35}$$

with a Gaussian one when performing inference. EP can be thought of as approximately minimizing $\mathrm{KL}[p(\mathbf{f}|X, Z)||q(\mathbf{f}|X, Z)]$ iteratively.

[11] In this work we apply a logistic logit sigmoid function, $\sigma(f(\boldsymbol{x})) = (1 + \exp(-f(\boldsymbol{x})))^{-1}$.

2.8.3 Simulation

To illustrate the approach, it is applied to simulate optimization of target values for two fictitious physical processes with three design variables, one known constraint

$$\sum_{i=1}^{3} x_i = 10, \tag{2.36}$$

and three unknown constraints

$$\begin{aligned} x_1^2 + 3^{x_2} &< 10 \\ 3 * x_1 + x_3^2 &< 10 \\ 3 * x_2 + x_3^2 &< 10. \end{aligned} \tag{2.37}$$

The processes are simulated by the Levy and the Dixon & Price functions (see eq. (2.25) and (2.26)). One random target vector was chosen from the $\left[f_{min}^{(1)}(\boldsymbol{x}), f_{max}^{(1)}(\boldsymbol{x})\right] \times \left[f_{min}^{(2)}(\boldsymbol{x}), f_{max}^{(2)}(\boldsymbol{x})\right]$ box, identified using 10000 feasible solutions. The set of candidate solutions, X_L, comprises of 2000 uniformly spread out input vectors satisfying (2.36) (the known constraint). The initial training set, X_T, comprises of 30 uniformly spread out input vectors and corresponding true labels (feasible/infeasible) obtained by interacting with the 'real system', which in our case amounts to checking the input vectors against *all* of the constraints (known and unknown). For input vectors, within the training set, identified as feasible the corresponding values of two processes are also obtained. For regression, a GP with zero mean and Squared Exponential (ARD) covariance function was employed. Process values are perturbed by noise drawn from $\mathcal{N}(0, 0.1^2)$. In (2.20), $k = 2|X_T|$ is used. The NSGA-II algorithm is iterated 100 times per iteration of the main algorithm with a population size at most 1% of $|X_L|$. For each mutation, the value of the perturbation was drawn from the uniform distribution on the interval $(0, \alpha 10]$. Values of the parameter α from the interval $[0.01, 0.1]$ where tested and $\alpha = 0.01$ selected. The decision set is re-sampled (as described in section 2.5), if there has been no improvement in the value of hypervolume indicator for three consecutive iterations. In step 3 of the adjusted MOAL algorithm the predicted feasible solutions are the solutions for which the sum of the predicted mean value and standard deviation of the class probability is above a threshold (values from the interval $[0.75, 1]$ were tested a value of 0.9 selected). The adjusted MOAL algorithm was run for 30 iterations. Figure 2.7 illustrates (in the solutions space) the progress made up to iteration 25. As can be seen from plots (c), (d) and (e), the efforts of the classification model, as intended,

are concentrated as requested by optimization. Consequently, the class boundary is identified accurately only where needed.

2.9 Conclusions

In this chapter a novel approach to multi-target optimization of expensive-to-evaluate functions based on the combined application of Gaussian Processes, Mutual Information and NSGA-II was presented. To illustrate the potential use of the approach, it was applied to simulate the optimization of target values of fictitious physical processes. Constrained and unconstrained optimization, using the proposed algorithm, was illustrated. The algorithm was compared against the surrogate based on-line Evolutionary Algorithm specifically designed for optimization of expensive-to-evaluate functions. Results indicate that, using the hypervolume indicator as a performance criterion, the proposed approach compares favourably against the surrogate based on-line Evolutionary Algorithm on tasks involving a small budget of evaluations. Although the computational complexity of the proposed algorithm is high, it is not foreseen to be a hindrance for the type of applications it is designed for. Additionally, we presented an adaptation of the MOAL algorithm for target optimization where some of the constraints are unknown. To illustrate the potential use of the approach, it was applied to simulate the optimization of target values of fictitious physical processes in the presence of unknown constraints. We appreciate that the MOAL algorithm, as presented in section 2.8, is yet to be tested on a number and variety of optimization problems involving unknown constraints.

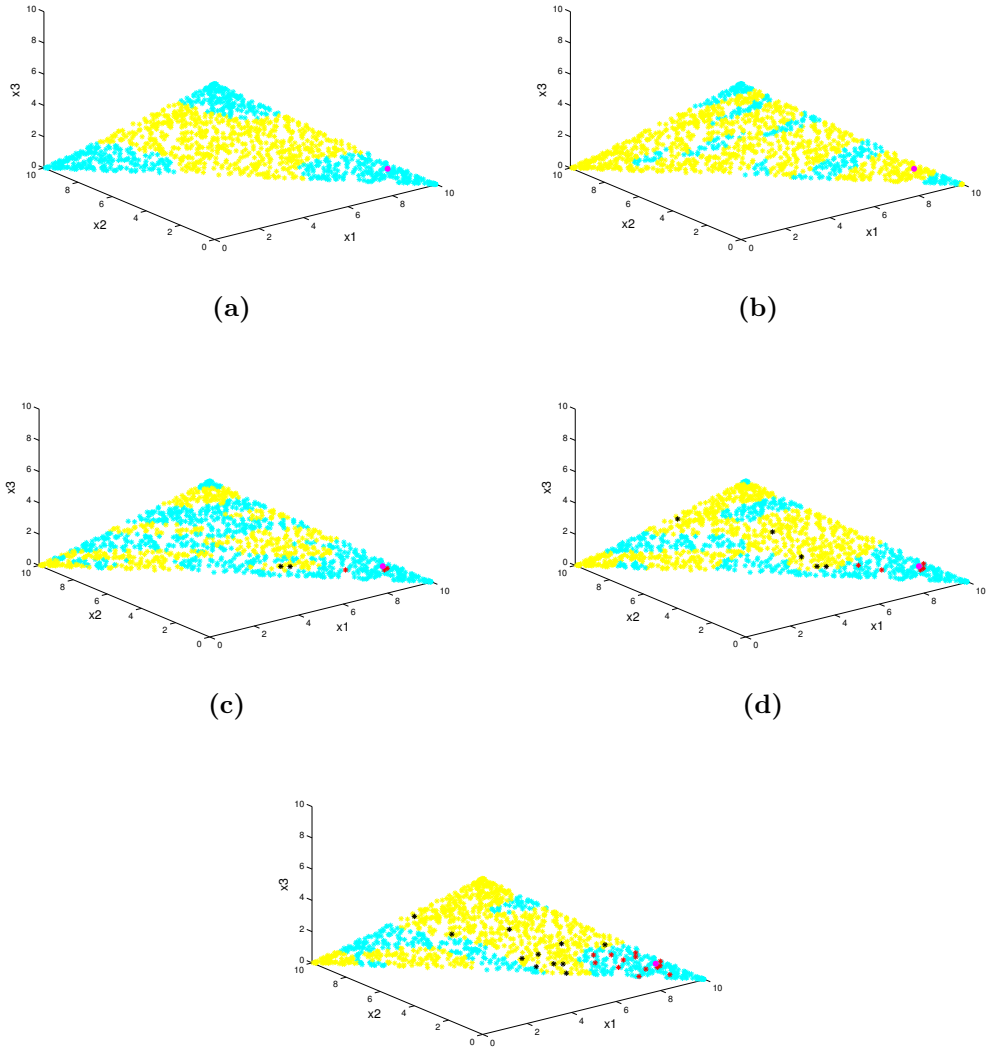


Figure 2.7: Simulated example of multi-target optimization where some constraints are unknown. Cayn stars - solutions that are predicted or evaluated as feasible; yellow stars - solutions that are predicted or evaluated as infeasible; red stars - sequentially evaluated solutions observed as feasible; black stars - sequentially evaluated solutions observed as infeasible; magenta filled circle - optimum solution (solution achieving the value of the hypervolume indicator equal to 100%). (a) Decision set with true labels; (b) decision set with labels predicted using the initial training set; (c) decision set with labels predicted after 5 iterations; (d) decision set with labels predicted after 10 iterations; (e) decision set with labels predicted after 24 iterations.

Chapter 3

Sequential Multi-Target Optimization of a Copolymerization Reaction

3.1 Brief summary of the chapter

In this chapter we consider optimization of a copolymerization reaction using the MOAL algorithm described in previous chapter. In particular, we describe on-line multi-target optimization of a copolymerization reaction where the requirement is to, in as few experiments as possible, find reaction conditions that: provide maximum conversion; produce, on average, polymer particles of 100 nm in diameter. Additionally, we consider the possibility of using the data acquired via the optimization (in particularly the set of optimal solutions) for pattern recognition with future similar optimization problems in mind.

3.2 Introduction

Polymers are molecules composed of a large number of many repeated subunits, called monomers. They have a number of useful properties [48] and as such are the building blocks of a lot of everyday items [49]. Polymers are made via a process called polymerization, where monomer molecules are reacted together in a chemical reaction to form polymer chains [50].

In polymerization, optimization of the underlying processes is crucial for establishing safe and cost-effective product manufacturing. Current polymerization pro-

cesses are run according to pre-defined recipes. A recipe includes information about starting concentrations of reactants (monomers), surfactant(s) and the reaction initiator, process temperature and likely duration of the reaction to completion. In semi-batch polymerizations a recipe is time-dependent as monomers are fed into the reactor over a period of time, thus maintaining a certain amount and concentration ratio, and thus also controlling the heat output of the reaction. In this case optimization is done typically off-line. Physical models, if available, allow for off-line fast optimization. However, their adequacy, in terms of parameter values, has to be considered. Recently, the so-called on-line estimation-optimization approaches have been investigated [51], where available process measurements are used to obtain reliable estimates of physical model parameters, as part of a closed loop optimization routine. All these models are optimizing for final product yield, but typically can not optimize for product quality.

Optimization of the underlying processes is also equally important for understanding the effects of the reaction mechanism and reaction conditions on the molecular and morphological properties of the product, as these are closely linked with the end-use properties [51]. This leads to multi-objective optimization of polymerization processes, i.e. the reaction/operating conditions sought have to be optimal with respect to both required product properties, product yield and cost of production.

The approaches described above are suitable for optimization of operating polymerization processes when the 'polymerization recipe' is known. A different type of optimization is the design-of-experiments approach when a new polymerization process is being developed and no information exists about the mechanism or about the optimal recipe. In this case, a significant number of experiments is required to develop the best recipe and collect the necessary information for developing a physical model that could later be used in real-time process optimization. In this chapter we describe the optimization of an emulsion copolymerization reaction. More specifically, we are required, in as few experiments as possible, to find reaction conditions for an assumed unknown process, that:

1. Provide maximum conversion.

2. Produce, on average, polymer particles of 100 nm in diameter.

It should be noted that there are usually also requirements of an auxiliary type, such as finding reaction conditions that minimize total reaction duration or cost of ingredients. An often practised approach is to carry out the optimization on the primary

objectives (objectives 1 and 2 in the above list) and then pick (from the identified set of Pareto solutions) the solution that is optimal in terms of the auxiliary objectives, i.e., in our case, solution with shortest total reaction duration and/or lowest total cost of ingredients. This is not part of the investigation at hand. However, references as to fitness of the obtained solutions in terms of auxiliary requirements will also be made.

Over time, via an ongoing research into polymerization processes, some intuition about the reaction conditions favourable to achieve a final product/process with a specified property, such as 100% conversion, for instance, has been built up. However, the knowledge/intuition for reaction conditions, when product/process properties are not considered in isolation, but rather simultaneously, is scarce and is often acquired through extensive experimentation. The factors, associated with experiments, that make such investigations expensive, are: time (individual experiments can often take a number of hours); material and human costs. Moreover, as, the required product/process properties' values vary from product to product, the associated experimentation is often conducted 'from scratch'. In this work we aim to, via application of the MOAL algorithm, solve the multi-target optimization problem at minimal cost (i.e. in a small number of experiments) and contribute to building up knowledge/intuition associated with solving problems of similar type.

In the next section we describe the experimental set up and the adaptation of the MOAL algorithm to optimization of a copolymerization process. In sections 3.4 and 3.5 we discuss the results of the optimization and provide conclusions respectively.

3.3 Methods

3.3.1 Experimental set up

Experiments are simulated by a computer model described in [54]. The model output is found to provide good agreement with observations obtained via laboratory experimentation and, hence, in this work, assumed to represent the 'unknown' underlying physical process. All parameters involved in the emulsion copolymerization reaction discussed herein are uniquely identifiable. The main assumptions and characteristics of the computer model are summarized as follows [54]:

- Semi-batch seeded copolymerization of a two-monomer system is the simulated process.

- The assumed chemical species are: water, initiator, monomer A, monomer B, a chain-transfer agent (CTA) and the produced copolymer.
- The emulsion is formally divided into three phases: (i) continuous aqueous phase, (ii) monomer droplets, and (iii) polymer particles.
- Polymerization is assumed to proceed only in the polymer particles (except the initiation stage, which proceeds in water).
- The kinetic model is terminal. Both inter-molecular and intra-molecular chain transfer (backbiting) are considered as well as the gel effect. Polymerization kinetics are processed by the population balance of polymer moments.
- The model describes stages II and III of the emulsion polymerization. Therefore, the total number of polymer particles in the reactor is considered constant during the simulation.
- The time-averaged number of radicals in the polymer particles is variable, calculated in the same way as in [53].
- Heat balance of the reaction mixture and of the reactor jacket is included in the model.
- All kinetic constants of the copolymerization reaction simulated by the model are based on data published in the literature.

Key model equations

The key model equations represent:

1. Algebraic equations
    - Evaluation of density of the droplet and the polymer phase as well as of the reaction mixture.
    - Summation relations (volume additivity of phases).
2. Differential equations
    - Mass balance of each monomer in each phase.
    - Total mass balance of each phase.

Total volume of reaction mixture V consists of the volumes of all present phases (Monomer droplets, Polymer particles and Water):

$$V = V^M + V^P + V^W. \tag{3.1}$$

Density of phase j is given by contributions of species i:

$$\frac{1}{\rho^j} = \sum_{i=1}^{k} \frac{w_i^j}{\rho_i} \tag{3.2}$$

with w_i^j being weight fraction of i-th species in phase j and ρ_i represents density of pure component i. Total number of components in a given phase is k. Note that monomer droplets consist only of monomers, while the polymer particles contain pure copolymer as well as absorbed monomer species. Density of aqueous phase is assumed to be constant.

Reaction mixture density is calculated from the overall mass balance of the reactor:

$$\frac{d\rho^{mix}}{dt} V + \rho^{mix} \frac{dV}{dt} = \dot{m}_F, \tag{3.3}$$

where $\dot{m}_F$ is the mass flowrate of the feed. Molar balance of monomer A in monomer droplets is implemented as

$$\frac{dc_A^M}{dt} V^M + c_A^M \frac{dV^M}{dt} = \frac{\dot{m}_F w_M^F w_A^{M,F}}{M_A} - (1 - stage3)\dot{n}_A. \tag{3.4}$$

The first term on the right hand side of the above equation represents the molar flowrate of monomer A in the feed with w_M^F being the weight fraction of the monomer phase in the feed, $w_A^{M,F}$ the weight fraction of monomer A in the monomer phase of the feed, and M_A is molar weight of monomer A. Dimensionless switching function $stage3$ is based on hyperbolic tangent, so that its value is 0 when the monomer droplets are present (transfer of monomers from droplets to polymer particles takes place) and 1 if the monomer droplets are depleted in this case the term representing the interfacial transport virtually 'disappears' from the model equation. This is a widely accepted method of modifying model equations during integration without the necessity to re-initialize the state variables and restart the integration. The form of switching function $stage3$ is

$$stage3 = 0.5 - 0.5 \tanh\left[10^{12}\left(V^M - V_{threshold}^M\right)\right], \tag{3.5}$$

with $V^M_{threshold} = 10^{-9}m^3$. The second term characterizes interfacial transport of monomer A from monomer droplets to polymer particles defined as

$$\dot{n_A} = K_A \left(w_A^{P^*} - w_A^P \right), \tag{3.6}$$

with K_A representing the mass transfer coefficient of monomer A, $w_A^{P^*}$ and w_A^P being the equilibrium and actual weight fraction of monomer A in the polymer phase.

Balance of monomer A in the polymer phase also has the term describing the transport of monomer into polymer particles, but with positive sign. Moreover, this equation must account for the consumption of monomer A by the chemical reaction R_A, which always has a negative absolute value:

$$\frac{dc_A^P}{dt}V^P + c_A^P\frac{dV^P}{dt} = (1 - stage3)\dot{n_A} + R_A V^P. \tag{3.7}$$

The model includes balance equations not only for chemical species, but also for all the present phases, which can be demonstrated with the total mass balance of monomer droplets

$$\frac{d\rho^M}{dt}V^M + \rho_A^M\frac{dV^M}{dt} = \dot{m_F}w_M^F - (1-stage3)\left(\dot{n_A}M_A + \dot{n_B}M_B + \dot{n}_{CTA}M_{CTA}\right). \tag{3.8}$$

Conversion is given by a ratio of the amount of reacted monomers and the amount fed into the reactor. Mean particle size is calculated by dividing volume of the polymer phase by number of particles present in the emulsion (which is a model parameter), assuming spherical shape of the particles.

The following are the reaction settings (input variables for the model considered):

- M_I^1, M_I^2, I_I, E_I, and W_I are the initial amounts of monomers, initiator, emulsifier and water respectively. T is the reaction temperature, CTA is the amount of chain transfer agent and P_0 is the amount of polymer in seed.
- M_F^1, M_F^2, E_F, I_F are the fed amounts of monomers, emulsifier and initiator respectively. WE_F and WI_F are emulsifier and initiator solutions in feed. WI_F is split into two parts pWI_F and $(1-p)WI_F$ (where $p \in (0,1)$) and is added in two stages.
- F_t^1, F_t^2 are feeding times for adding pWI_F and $(1-p)WI_F$ respectively and P_t is the post feeding time. The total reaction duration is the sum of F_t^1, F_t^2 and P_t.

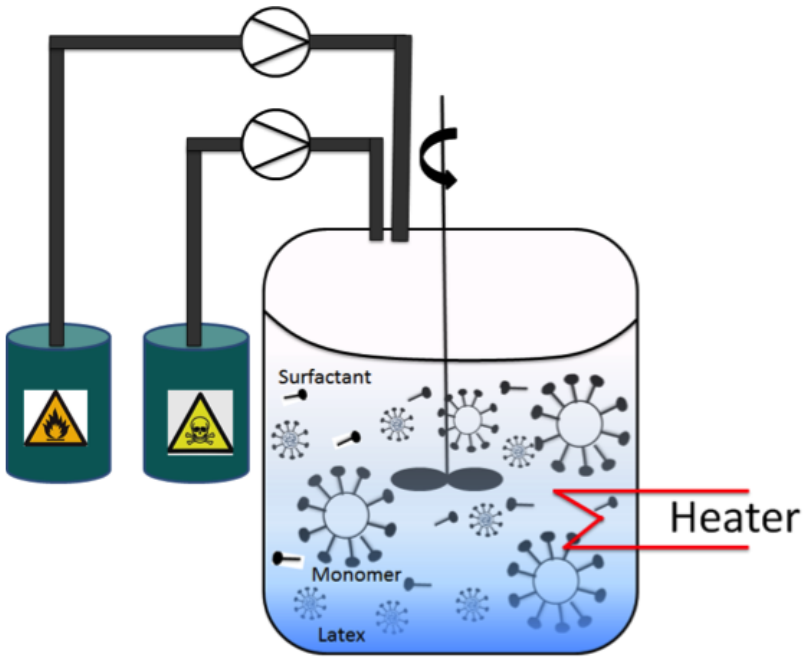


Figure 3.1: Emulsion copolymerization process scheme example.

The ranges of the variables and constraints are:

1. $M_I^1 = M_I^2 = 1 \times 10^{-10}$g and $E_I = E_F = 1 \times 10^{-4}$g. These are kept constant for all recipes.
2. W_I,WE_F, $WI_F \in (0, 1500\text{g}]$ and $225 \leq W_I + WE_F + WI_F \leq 1500$.
3. $I_I \in (0, 0.2W_I)$ and $I_F \in (0, 0.2WI_F)$.
4. F_t^1, F_t^2, $P_t \in (0, 180\text{min}]$ and $30 \leq F_t^1 + F_t^2 \leq 180$.
5. $T \in [65, 95]$ in degrees Celsius, $CTA \in (0, 20\text{g}]$ and $P_0 \in [5, 30\text{g}]$.
6. M_F^1, $M_F^2 \in (0, 1500\text{g}]$ and $150 \leq M_F^1 + M_F^2 \leq 1500$.
7. The final constraint is that the overall volume of the ingredients must not exceed the capacity of the reactor (3l).

3.3.2 Adaptation of the MOAL algorithm

Our problem consists of fourteen input dimensions, four constraints and two (primary) objectives. In order to apply the MOAL algorithm, we convert all of the

Figure 3.2: Example of a laboratory scale reactor that would be used in the emulsion copolymerization reaction experimentation. The capacity of this reactor is 0.5l.

objectives into targets. Thus the first objective becomes a target of '100% conversion'. The target vector is, thus, $[100\ 100]^T$. The set of candidate solutions, X_L, comprises of 2000 input vectors uniformly spread out over the decision space. The initial training set, X_T, comprises of 15 input vectors, obtained as a Latin Hyper Cube (LHC) sample, and correspondingly computed, using the computer model, values of conversion and average particle diameter. The size of the initial training set reflects the intuition relayed by the chemical engineers familiar with the process. Namely, even though there are fourteen input variables involved, only a few are expected to influence the response variables. A GP with zero mean function and $\text{Matérn}_{\nu=\frac{3}{2}}$ covariance function is employed to construct the surrogate models for approximation of both conversion and average particle diameter. Further details of the set up are:

- Process values are perturbed by noise drawn from $\mathcal{N}(0, 0.03^2)$. The noise value was chosen on the advice of the chemical engineers familiar with the process.
- The input variables are normalized (using the allocated ranges) to be in the $[0, 1]$ interval, prior to application of the surrogate models.
- In (2.20), $k = 2|X_T|$ is used.
- The NSGA-II algorithm is iterated 100 times per iteration of the main algorithm with a population size at most 1% of $|X_L|$. For each mutation, the value of the perturbation is drawn from the uniform distribution on the interval $(0, \alpha 1]$, where $\alpha = 0.01$ is used. The values chosen for the parameters α and k (from previous bullet point) are the same as the ones used in simulations in the previous chapter. This is because: these values produced good results in simulations; and we only allow for one optimisation run for the copolymerisation reaction problem (as would often be the case with real laboratory experiments).
- The decision set is re-sampled, if there has been no improvement in the value of the hypervolume indicator for 20 consecutive iterations. To re-sample, only the solutions so far collected are considered. These solutions are assumed to belong to a mixture of Gaussian distributions with fourteen components. A Gaussian mixture model is fitted and the parameter estimates (components' means, covariances and mixture proportions) are obtained using an Expectation Maximization (EM) algorithm. 2000 input vectors are then drawn from the resulting distribution.

The hyperparameters of the surrogate models are fitted by optimizing the marginal likelihood using a conjugate gradient optimizer. To avoid bad local minima five random restarts are tried, picking the run with the best marginal likelihood. Leave-one-out cross-validation is used to validate the models. Namely, for each point in the training set, its predicted function value along with the variance of the predicted value are computed using the rest of the set. Cross-validated standardized residuals (see (2.28)) are computed and a check is carried out that, the standardized residuals are all in the interval $[-3, +3]$ (following [33]). The optimization is run for 70 iterations. After the last iteration, from all of the solutions collected (including the initial training set), the optimal solutions are identified as the ones for which the corresponding observations satisfy

$$\mathrm{C} \geq 99\%, \tag{3.9}$$

and

$$99 \leq \mathrm{D_{avg}} \leq 101, \tag{3.10}$$

where $\mathrm{D_{avg}}$ is average polymer particle diameter and C is conversion.

3.4 Results and discussion

Running the optimization using the MOAL algorithm has provided the following results:

1. Optimization resulted in a set of 18 optimal solutions satisfying inequalities (3.9) and (3.10). This indicates that the optimisation problem is highly multimodal.

2. The earliest experiment yielding an optimal solution was experiment 24. The reaction duration used in this solution was 197 min. Note, that it is often acceptable to stop the optimisation as soon as one optimal solution, satisfying inequalities (3.9) and (3.10), is found. This would bring the total of the experiments to just 39 (to include the 15 experiments from the initial training set).

3. The shortest reaction duration (among the 18 solutions) was 169 minutes (corresponding to experiment 80).

The following observations, based on the 18 optimal solutions, are made:

- The average total amount of monomers fed in is 368 g with a standard deviation of 7 g.
- Looking at the amount of initiator in feed and post feeding time data, it can be suggested that there are (at least) two modes of operation (see Figures 3.3 and 3.4). One mode consists of solutions with small amounts of initiator. This could be exploited, if, for instance, the cost of ingredients was included as one of the auxiliary objectives, as the initiator is often one of the more expensive ingredients. However, it should be noted that this observation is based on the sample of data collected via the optimisation run (not through exploratory design of experiments), and, hence, should be treated with caution.

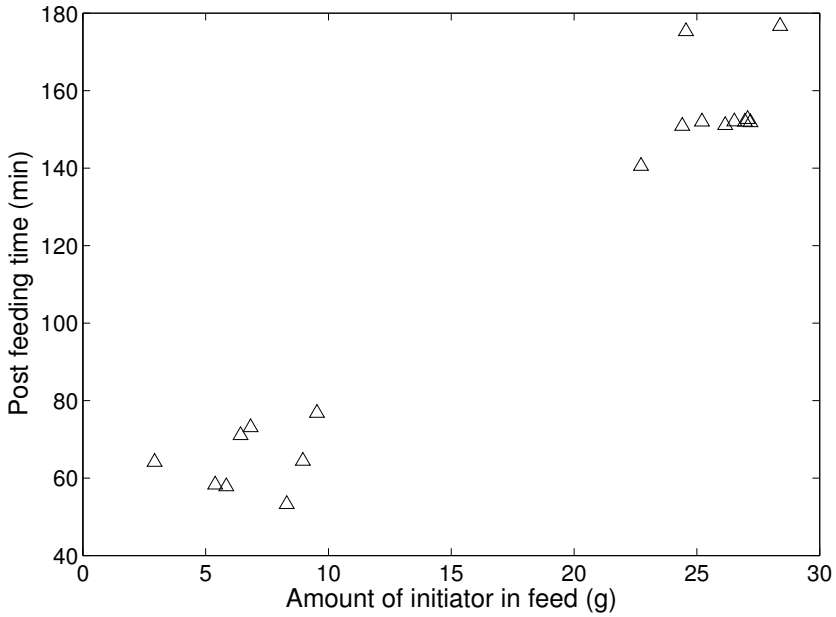


Figure 3.3: Amount of initiator in feed vs. post feeding time for the 18 optimal solutions.

Following the above observations, we investigate the effect of varying the total amount of monomers (across the allowable range) on conversion and the average particle diameter, using the identified optimal solutions otherwise unchanged. Namely, we, first, select one optimal solution (the earliest identified solution is used), secondly, vary the total amount of monomers in this solution (keeping the ratio of the monomers as well as the values of the rest of the variables unchanged and checking that none of the constraints are violated), and finally, evaluate thus designed solutions, using the computer model and the surrogate models as trained via the

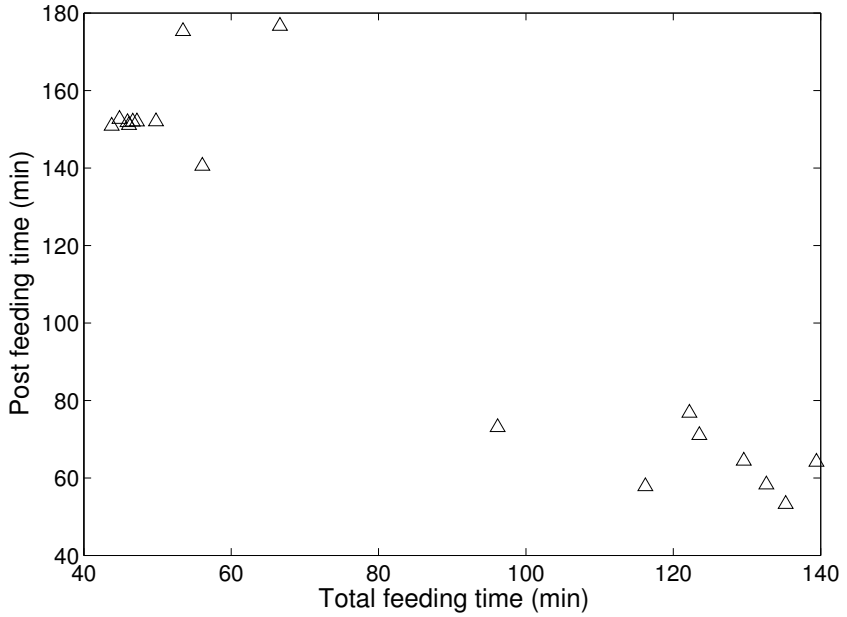


Figure 3.4: Total feeding time ($F_t^1 + F_t^2$) vs. post feeding time for the 18 optimal solutions.

optimization, to obtain corresponding values of conversion and average particle diameter (see Figure 3.5 and 3.6). As can be seen from Figures 3.5 and 3.6, the surrogate models as trained via the optimization make poor approximations for solutions far away from the selected optimal one. This can be explained by the fact that the training set was designed (bar the initial 15 points) to ensure optimal exploration with target values for conversion and average particle diameter in mind, and, as such, is less effective for the global approximation of conversion and average particle diameter. However, the situation is, easily, corrected by evaluating a further, small number of solutions this time specifically designed to optimally explore the identified hyperplane of the 14 dimensional space. To do that, we generate 2000 candidate solutions uniformly spread out around the hyperplane and then run the optimization for a further 15 iterations with the currently available training set as the initial training set and exploitation 'switched off'. Namely, at each iteration of the MOAL algorithm, we select the solution that is optimal according to (2.17), where $\|\boldsymbol{r}(\boldsymbol{x}_j)\|$ is set to 0. Then, the surrogate models are trained on the updated training set and used for prediction again (see Figures 3.7 and 3.8). As we are optimising real physical processes using expensive evaluations, it is important that the number of further explorative iterations is kept to a minimum. However, at present, we are not aware of a robust stopping rule to affect this problem. Also, due

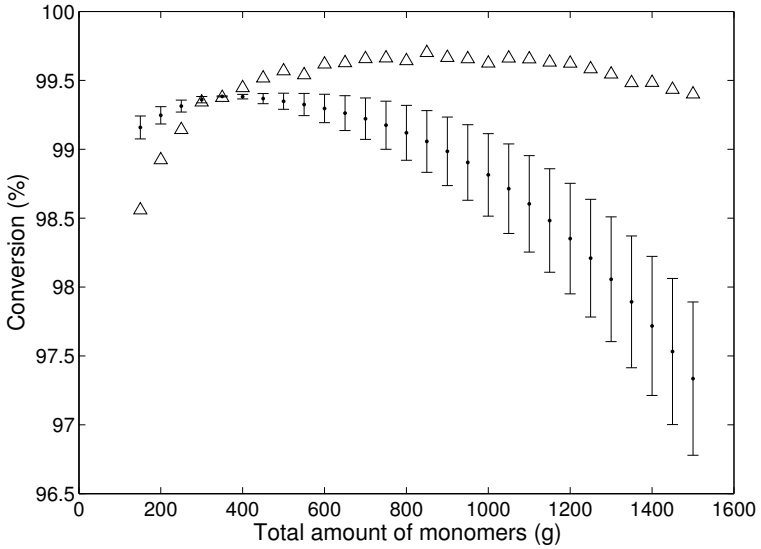


Figure 3.5: Conversion values, computed using the computer model (triangles) and estimated mean conversion values (dots) along with the 2 standard-deviation error bars, predicted using a surrogate model, for one of the optimal solutions with varied total amount of monomers.The surrogate model is trained using the training set available immediately after target value optimization.

to the fact that in our scenario only one optimisation run is budgeted for, we did not experiment with/investigate the design of novel automatic stopping rules.

As can be seen from Figures 3.7 and 3.8, approximate values of conversion and average particle diameter obtained using the surrogate models trained on the further updated training set are in good agreement with the corresponding values obtained using the computer model. We note that all of the predicted/computed conversion values in Figure 3.7 are above 98.5% with most being above 99%. Thus, for the experimental set up as described in this chapter, a solution needed to achieve (at worst) 99% conversion and any required value of average particle diameter (up to ± 1 nm error) in the range of 85 to 160 nm (or 75 to 160 nm, if inequality (3.9) is relaxed to $C \geq 98.5\%$) can now be estimated. With regards to the aims set out to be achieved at the start of the investigation, we observe that, via application of the MOAL algorithm:

- The multi-target optimization problem, as set out in this chapter, can be solved at a modest cost. The smallest total number of (simulated) expensive evaluations needed to find an optimal solution was 39. These are made up of

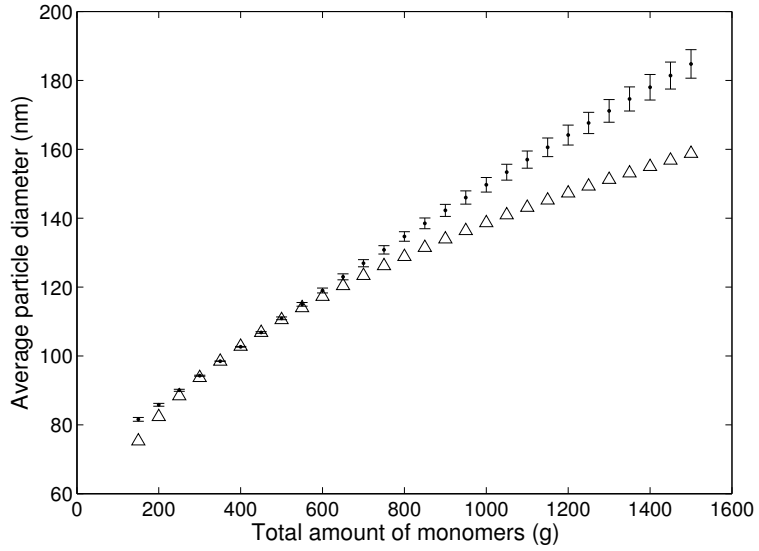


Figure 3.6: Average particle diameter values, computed using the computer model (triangles) and estimated mean average particle diameter values (dots) along with the 2 standard-deviation error bars, predicted using a surrogate model, for one of the optimal solutions with varied total amount of monomers. The surrogate model is trained using the training set available immediately after target value optimization.

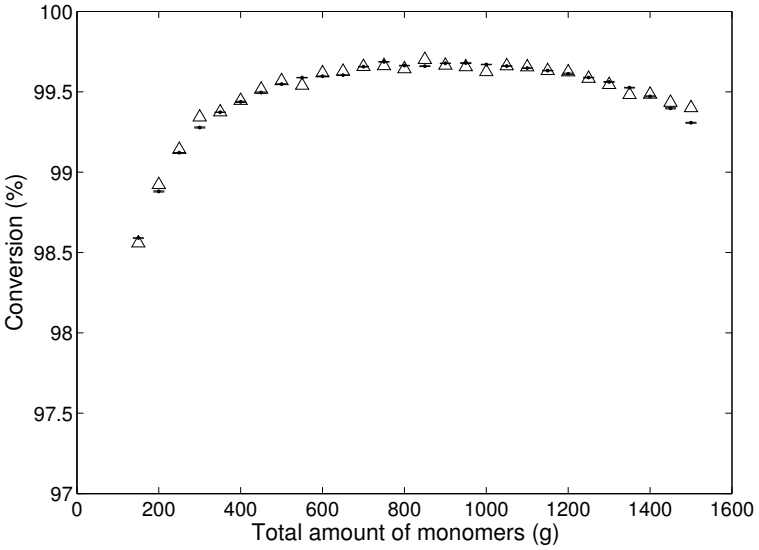


Figure 3.7: Conversion values, computed using the computer model (triangles) and estimated mean conversion values (dots) along with the 2 standard-deviation error bars, predicted using a surrogate model, for one of the optimal solutions with varied total amount of monomers. The surrogate model is trained using further updated training set.

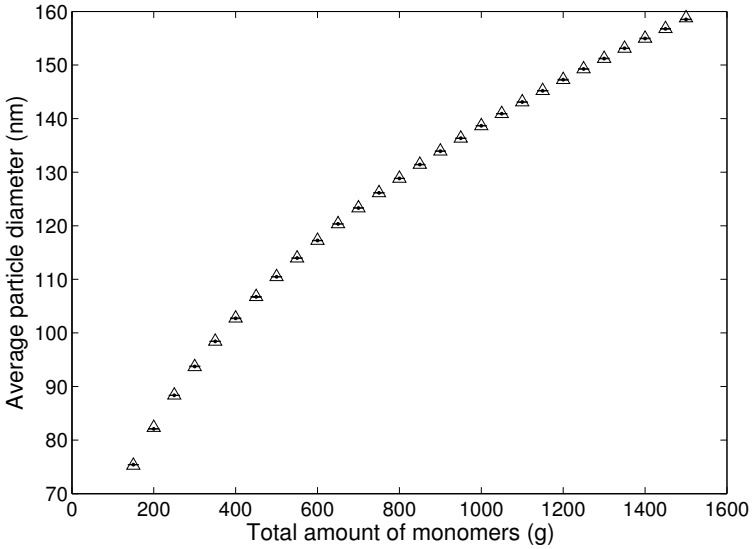


Figure 3.8: Average particle diameter values, computed using the computer model (triangles) and estimated mean average particle diameter values (dots) along with the 2 standard-deviation error bars, predicted using a surrogate model, for one of the optimal solutions with varied total amount of monomers. The surrogate model is trained using further updated training set.

15 initial evaluations and 24 evaluations needed to identify the first optimal solution. The average typical computational time taken by the MOAl algorithm per iteration was 40 seconds, with the first optimal solution found in 16 minutes.

- An intuition was acquired, albeit only restricted to the type of problems as set out in this chapter, for an approach that allows the identification of solutions associated with a *range* of possible target values of average particle diameter (and conversion of (at worst) 99%). However, this resulted in additional cost. Namely, the number of (simulated) expensive evaluations performed in this case totalled 100. We speculate (and intend to work on it as a future extension of the MOAL algorithm) that this additional cost could, potentially, be minimised through collecting the necessary local information during the overall optimisation run of the MOAL algorithm.

3.5 Conclusions

In this chapter we considered optimization of a copolymerization reaction using the MOAL algorithm described in chapter 2. In particular, we described on-line multi-target optimization of a copolymerization reaction where the requirement was to, in as few experiments as possible, find reaction conditions that: provide maximum conversion; produce, on average, polymer particles of 100 nm in diameter. Additionally, we considered the possibility of using the data acquired via the optimization (in particular the set of optimal solutions) for pattern recognition with future similar optimization problems in mind.

Using the MOAL algorithm, we were able to find an optimal solution within a modest number (39) of expensive evaluations. Analysis of optimal solutions obtained via optimization has led to the discovery of a variety of different modes of optimal solutions. We were also able to detect an input-output variable dependency. Namely, for an optimal solution, the average particle diameter was found to vary non-linearly with the total amount of monomers fed in (given all other input variables' values and the ratio of the monomers are kept unchanged and none of the constraints are violated). The non-linearity of the dependency, however, is not unexpected, given the chemical reaction system under consideration. It was also confirmed that varying the total amount of monomers in the optimal solution had little to no effect on the value of conversion. These findings can be employed (for the type of problems as set out in this chapter) in identification of solutions associated with a *range* of possible target values of average particle diameter (and conversion of (at worst) 99%). For the post optimization investigation, to insure satisfactory predictive ability of the surrogate models in the relevant part of the decision space, 15 extra evaluations were performed, taking the total number of evaluations to 100. We, however, consider this additional cost worthwhile.

Chapter 4

Application of dimensionality reduction

4.1 Brief summary of the chapter

Several dimensionality reduction techniques were applied to two data sets of consumer products formulations in order to infer their intrinsic structure and specific product design rules. Data sets of sufficient size, obtained via high throughput experimentation, were used. The Supervised Isometric Feature Mapping (S-Isomap) was combined with a k-Nearest Neighbours (k-NN) classifier and k-means clustering algorithm to perform categorization of viscosity of new formulations, not used to train the model. We compared prediction results of this approach with several well-established classification models. The results show the accuracy of the S-Isomap based approach to be superior and with a potential for further improvement. Compared with other dimensionality reduction techniques, applying the S-Isomap has allowed for a superior visualization of category separation within the formulations, for the data sets used.

4.2 Introduction

The aim of chemical product design is to find a product that exhibits certain specified behaviour/properties, corresponding to the desired functional properties. Thus, in the area of formulated consumer products the main useful functions, for example the function of 'moisturising' or the function of 'UV protection', are achieved through the use of molecules and particles with corresponding physico-chemical properties, e.g., UV absorbance and scattering of $Ti0_2$ micro-particles reducing the flux of the

harmful range of UV solar radiation to the skin. These main useful functions in consumer products are accompanied by a varied cocktail of secondary desired functions, such as 'feel', 'smell', 'colour', etc.

Several important technical challenges in the design of formulated products stem from the fuzziness of performance criteria for a number of desired functions, not easily converted to numerical specifications, and their apparent complexity, which does not allow easy prediction of properties based on composition. The latter means that the performance property is often described as a 'system property': a property that emerges due to interactions among individual components of a system i.e., the ingredients of a formulated product in our case. Viscosity is one example of such a system property within the context of formulated consumer products: it is often necessary for a formulated product to have a particular rheological behaviour e.g., shear thinning behaviour or a Newtonian liquid behaviour, which results in a particular 'feel' function. However, our capacity to accurately predict viscosity of formulated consumer products based on composition is quite limited, due to the physical complexity of the system. Thus, we can describe viscosity as a system property, not equal to the linear combination of the contributions of the ingredients used. Viscosity is used in the present study as the output performance criterion for characterisation of the formulations. We should note, however, that viscosity is not a true 'emergent' property, since its apparent complexity stems from our inability to accurately describe the physico-chemical phenomena involved. This is a common problem in multi-component engineered systems and thus the developed tools presented in this thesis have much wider applicability than the example of consumer products design used.

In the design of formulations, the data used for model development may be of moderate to high dimensionality, which is prohibitive to visualisation and hence to simple methods of interpretation. Dimensionality reduction techniques may simplify visualisation of multi-dimensional data, thus allowing for learning about the hidden structure of the data within high/moderate dimensional space to take place. Also, using a dimensionality reduction technique prior to application of a classification algorithm can, potentially, help make more accurate predictions.

There are a number of techniques that have been developed for dimensionality reduction. The widely known and used ones are Principal Component Analysis (PCA) and multidimensional scaling (MDS). The goal of PCA is to compute the

most meaningful basis with which to re-express a data set. The hope is that the obtained new basis will filter out the noise and reveal the hidden structure in the original data. PCA is defined as an orthogonal linear transformation that transforms the data to a new coordinate system in such a way that the greatest variance by any projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on [63].The data used for MDS are dissimilarities between pairs of objects. The main objective of MDS is to represent these dissimilarities as distances between points in a low dimensional space such that the distances correspond as closely as possible to the dissimilarities. However, the effectiveness of both PCA and MDS is limited by their global linearity. In order to resolve the problem of dimensionality reduction in nonlinear cases, manifold learning techniques such as locally linear embedding (LLE) [64] and the Isomap [65], for instance, have been proposed.

There is a variety and a number of dimensionality reduction techniques available for application. In our problem, as we do not have any knowledge of the underlying process and due to a limited amount of time allocated to working on this particular application[1], we restrict ourselves to testing a small number of techniques. For visualisation, we use three non-linear dimensionality reduction techniques:

- Locally Linear Embedding(LLE)
- Isometric Feature Mapping (Isomap)
- Supervised Isometric Feature Mapping (S-Isomap)[66]

and two linear techniques:

- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA) [67].

For classification we use the S-Isomap. Varieties of implementations of the Supervised Isomap have been successfully used in many problems that required non-linear dimensionality reduction prior to classification [57; 58; 59].

[1]This particular application was one of the deliverables proposed/outlined for completion by the end of the 3 year PhD project and, as such, the time allocated for it amounted to a fraction of 3 years (6 months, to be more precise).

4.3 Experimental

4.3.1 Data sets

Two data sets were provided by Unilever plc. The data is a set of complex mixtures (described by formulations) with a categorisable system property of the resulting product - viscosity. The idea is to use this set of experimental data to create a model/classifier which can then be used to predict the viscosity category of mixtures described by 'unseen' formulations. The ingredients used are typical of those employed in designs of consumer formulated products, such as detergents for instance.

The first data set was used for visualisation and classification. The second dataset was used for visualisation only. The first dataset consisted of 170 formulations. Each formulation was described as:

- A combination of proportions of five ingredients: *clay*, *biomass*, *surfactant*, *salt* and *water*.
- Two intrinsic properties: *acidity of clay* and *acidity of the final product*.
- A description of how viscous the final product is: *strong gel*, *medium gel*, *weak gel*, *high viscosity liquid*, *medium viscosity liquid*, *low viscosity liquid*. The descriptions were obtained by visual observation by one experienced formulator, and thus assignment of descriptors is consistent within the series. Qualitative descriptors were used to mimic user perception of a formulated product and to speed up the high-throughput experimentation.

Since there are five ingredients and two intrinsic properties per formulation, we assumed each formulation to be a point in a seven-dimensional space. The seventh attribute, namely *acidity of the final product*, does not add to the variation within the data due to strong correlation with the *acidity of clay* attribute. Therefore we reduced the number of attributes to six. The *high viscosity liquid* category was only represented by four observations. Hence, the four observations were excluded from the data set prior to classification.

The second dataset consisted of 55 formulations. Each formulation was described as:

- A combination of proportions of three ingredients: *biomass*, *surfactant* and *water*.

- Two intrinsic properties: *acidity of the final product* and *the point of highest curvature* from the phase angle vs. the oscillation stress curve. The point of highest curvature was determined the following way: firstly, a curve was fitted to the phase angle vs. oscillation stress data using smoothing splines (a smoothing parameter was chosen experimentally for each curve and goodness of fit was evaluated graphically and numerically (using the sum of squares due to error (SSE) statistic)); then, the second derivative of the fitted curve was computed and the point with highest absolute value selected.
- A description of how viscous the final product is: *strong gel*, *medium gel*, *weak gel*, *high viscosity liquid*, *medium viscosity liquid*, *low viscosity liquid*.

We assumed each formulation in the second data set to be a point in a six-dimensional space.

4.4 Methods

4.4.1 Dimensionality Reduction

The aim is to map a data set $X = [\boldsymbol{x}_1, ..., \boldsymbol{x}_N]$, $\boldsymbol{x}_i \in R^m$ to a data set $Y = [\boldsymbol{y}_1, ..., \boldsymbol{y}_N]$, $\boldsymbol{y}_i \in R^d$ with $d < m$, while retaining the geometry of the data as much as possible.

Unsupervised Dimensionality Reduction Techniques

LLE

In LLE, the mapping to a low-dimensional space is constructed by considering the symmetries of locally linear reconstructions. The idea is that, given enough data, each data vector can be replaced by a linear combination of the k nearest other ones, leading to the problem of finding reconstruction weights that minimise the error function:

$$\xi(\boldsymbol{W}|X) = \sum_{i=1}^{N} \|\boldsymbol{x}_i - \sum_{j=1}^{N} \boldsymbol{W}_{ij}\boldsymbol{x}_j\|^2, \tag{4.1}$$

where $\boldsymbol{W}_{ij}$ are the unknowns, subject to ($\boldsymbol{W}_{ij} \neq 0$ only for the k closest neighbours of each point) and

$$\sum_{j=1}^{N} \boldsymbol{W}_{ij} = 1. \tag{4.2}$$

The $\boldsymbol{W}_{ij}$ are then determined by solving a set of constrained least squares problems. The idea is that the $\boldsymbol{W}_{ij}$ convey the intrinsic geometric properties of the data that are expected to be valid, locally, in the reduced space. Given $\boldsymbol{W}$ is computed, a new error function, where the unknowns are now the low-dimensional coordinates $\boldsymbol{y}_i$ associated with each $\boldsymbol{x}_i$, can be written as:

$$\phi(Y|\boldsymbol{W}) = \sum_{i=1}^{N} \|\boldsymbol{y}_i - \sum_{j=1}^{N} \boldsymbol{W}_{ij}\boldsymbol{y}_j\|^2, \tag{4.3}$$

subject to:

- $\sum_i \boldsymbol{y}_i = \boldsymbol{0}$, i.e. the output data are centred.
- The covariance of the output is a unit matrix [68], i.e. the new coordinates are uncorrelated.

Obtaining $\boldsymbol{y}_i$, using (4.3), is equivalent to carrying out an eigendecomposition of the matrix $(\boldsymbol{I}-\boldsymbol{W})^T(\boldsymbol{I}-\boldsymbol{W})$ and disposing of the eigenvector corresponding to the smallest eigenvalue, followed by taking the eigenvectors that correspond to the next (lower) eigenvalues.

The Isomap

The ISOMAP is an extension of MDS, where pairwise Euclidean distances between data points are substituted by the respective geodesic distances, computed by *graph shortest path distances*. Key steps are:

1. Construct a neighbourhood graph $G = (V, E, W)$ such that:
   - $V = \{\boldsymbol{x}_i : i = 1, ..., N\}$
   - $E = \{(\boldsymbol{x}_i, \boldsymbol{x}_j) : \text{if } \boldsymbol{x}_j \text{ is a neighbour of } \boldsymbol{x}_i\}$, i.e. $\boldsymbol{x}_j$ is one of the K-nearest neighbours of $\boldsymbol{x}_i$ or if they are closer than a certain distance ϵ
   - $W_{ij} = d(\boldsymbol{x}_i, \boldsymbol{x}_j)$, edge length (in Euclidean distance)
2. Compute graph shortest path distances (applying Floyd's algorithm). Initialize $d_G(x_i, x_j) = d(x_i, x_j)$ if x_i and x_j are linked by an edge, assign $d_G(x_i, x_j) = +\infty$ otherwise. Then, for each value of $k = 1, 2, ..., N$ in turn, replace all entries $d_G(x_i, x_j)$ by
   $\min\{d_G(x_i, x_j), d_G(x_i, x_k) + d_G(x_k, x_j)\}$.

3. Apply classical MDS with $D = (d_G(x_i, x_j)^2)$ (the matrix of squared distances). Namely:

   - Construct a symmetric $B = -0.5HDH^T$, where H is a 'centering' matrix ($H_{ij} = \delta_{ij} - 1/N$)
   - Find the eigenvector decomposition of $B = U\Lambda U^T$ and choose the top d eigenvectors
   - Obtain new dimensions as

     $$y_i^p = \sqrt{\lambda_p} u_p^i, \;\; p = 1, ..., d, \; i = 1, ..., N,$$

     where u_p^i is the ith component of the pth eigenvector and λ_p is the corresponding (to the pth eigenvector) eigenvalue

<u>*PCA*</u>

Principal component analysis (PCA) is a mathematical technique that transforms a number of (possibly) correlated variables $x^{(k)} \in \boldsymbol{x}, k = 1, ..., m$ into a (smaller) number of uncorrelated variables $y^{(k)} \in \boldsymbol{y}, k = 1, ..., d,$. The first principal component $\boldsymbol{\omega}_1$ is such that the input data, after projection on to $\boldsymbol{\omega}_1$, is most spread out, subject to $\|\boldsymbol{\omega}_1\| = 1$ (for a unique solution). Let $\boldsymbol{\Sigma}$ be the covariance[2] matrix[3] of $\boldsymbol{x}$, $V_{y^{(1)}}$ be the variance of $y^{(1)}$, and $y^{(1)} = \boldsymbol{\omega}_1^T \boldsymbol{x}$. Then

$$V_{y^{(1)}} = \boldsymbol{\omega}_1^T \Sigma \boldsymbol{\omega}_1,$$

and we seek $\boldsymbol{\omega}_1$ such that $V_{y^{(1)}}$ is maximised, subject to $\boldsymbol{\omega}_1^T \boldsymbol{\omega}_1 = 1$. Applying the method of Lagrange multipliers, we wish to find

$$\max_{\boldsymbol{\omega}_1} \left(\boldsymbol{\omega}_1^T \Sigma \boldsymbol{\omega}_1 - \alpha \left(\boldsymbol{\omega}_1^T \boldsymbol{\omega}_1 - 1 \right) \right)$$

Taking the derivative with respect to $\boldsymbol{\omega}_1$ and setting it to 0 leads to

$$2\Sigma\boldsymbol{\omega}_1 - 2\alpha\boldsymbol{\omega}_1 = 0,$$

and, further, to

$$\Sigma\boldsymbol{\omega}_1 = \alpha\boldsymbol{\omega}_1,$$

[2]Note, that using a correlation matrix instead may be more appropriate in situations where the variances of the original dimensions vary considerably.

[3]We assume $\boldsymbol{x} \sim \mathcal{N}_m(\boldsymbol{0}, \boldsymbol{\Sigma})$.

which is true if $\boldsymbol{\omega}_1$ is an eigenvector of Σ and α is the corresponding eigenvalue. Since the aim is to maximise

$$V_{y^{(1)}} = \boldsymbol{\omega}_1^T \Sigma \boldsymbol{\omega}_1 = \alpha \boldsymbol{\omega}_1^T \boldsymbol{\omega}_1 = \alpha,$$

the eigenvector with the largest eigenvalue is chosen. The second principal component $\boldsymbol{\omega}_2$ is obtained using the same routine as for $\boldsymbol{\omega}_1$, subject to a constraint that $\boldsymbol{\omega}_2$ is orthogonal to $\boldsymbol{\omega}_1$ (to ensure that $y^{(2)}$ is uncorrelated with $y^{(1)}$). The outcome is that $\boldsymbol{\omega}_2$ is the eigenvector of Σ with the second largest eigenvalue. Similarly, it can be shown that other dimensions are given by the eigenvectors corresponding to decreasing eigenvalues.

Supervised Dimensionality Reduction Techniques

LDA

Let the matrix χ be an m-dimensional set of N samples in a c-class problem. The set χ is partitioned into c subsets $\chi_1, \chi_2, ..., \chi_c$ where each subset χ_i belongs to a particular class and consists of N_i number of samples such that:

$$N = \sum_{i=1}^{c} N_i$$

We want to find the matrix $\boldsymbol{W}$ such that

$$\boldsymbol{y} = \boldsymbol{W}^T \boldsymbol{x},$$

where $\boldsymbol{y}$ is d-dimensional and $\boldsymbol{W}$ is $m \times d$. Given the sample set χ, the within-class scatter matrix (S_W) and the between-class scatter matrix (S_B) can be defined as

$$S_W = \sum_{j=1}^{c} \sum_{\boldsymbol{x} \in \chi_j} (\boldsymbol{x} - \mu_{\boldsymbol{x}}^{(j)})(\boldsymbol{x} - \mu_{\boldsymbol{x}}^{(j)})^T,$$

$$S_B = \sum_{j=1}^{c} N_j (\mu_{\boldsymbol{x}}^{(j)} - \mu_{\boldsymbol{x}})(\mu_{\boldsymbol{x}}^{(j)} - \mu_{\boldsymbol{x}})^T,$$

where $\mu_{\boldsymbol{x}}^{(j)}$ is the centroid of χ_j and $\mu_{\boldsymbol{x}}$ is the centroid of χ. After projection, the between-class scatter matrix and the within-class matrix are $\boldsymbol{W}^T S_B \boldsymbol{W}$ and $\boldsymbol{W}^T S_W \boldsymbol{W}$ respectively, and are both $d \times d$. We want the between-class scatter to

be large, as we want the class means to be as far apart from each other as possible. And we want the within-class scatter to be small, as we want the samples from the same class to be as close to their mean as possible. Hence we want matrix $\boldsymbol{W}$ that maximises the ratio

$$J(\boldsymbol{W}) = \frac{|\boldsymbol{W}^T S_B \boldsymbol{W}|}{|\boldsymbol{W}^T S_W \boldsymbol{W}|},$$

where $|\bullet|$ denotes the determinant[4]. $\boldsymbol{W}$ is the solution of the eigenvalue problem

$$S_W^{-1} S_B \boldsymbol{w}_i = \lambda_i \boldsymbol{w}_i, \tag{4.4}$$

where $\boldsymbol{w}_i$ are the column vectors of $\boldsymbol{W}$. The desired d leading eigenvectors are selected for $\boldsymbol{W}$ that corresponds to the largest eigenvalues λ_i in Eq.(4.4).

S-Isomap

In the S-Isomap, for the given observations $(\boldsymbol{x}_i, z_i)$, $i = 1, ..., N$, where $\boldsymbol{x}_i$ is the input vector and z_i is its class label, the dissimilarity, D, between two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is defined as [66]:

$$D(\boldsymbol{x}_i, \boldsymbol{x}_j) = \begin{cases} \sqrt{1 - e^{\frac{-d^2(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\beta}}}, & z_i = z_j \\ \sqrt{e^{\frac{d^2(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\beta}}} - \alpha, & z_i \neq z_j \end{cases} \tag{4.5}$$

Where $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ denotes the Euclidean distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. Since the Euclidean distance $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is in the exponent, the parameter β is used to prevent $D(\boldsymbol{x}_i, \boldsymbol{x}_j)$ from increasing too fast when $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is relatively large. Thus, the value of β should depend on the 'density' of the data set. Usually, β is set to be the average Euclidean distance between all pairs of data points. The parameter α gives a certain chance to the points in different classes to be 'more similar' i.e., to have a smaller value of dissimilarity, than those in the same class (for a more in-depth discussion on the choice of the value for α see [66]). The following steps are used in the S-Isomap:

1. Construct the neighbourhood graph of the input data, according to the dissimilarity between data points (the neighbourhood could be the k most similar points or the points whose dissimilarity is less than a certain value ϵ). Connect the two neighbouring points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ with an edge, weighted as $D(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

[4]As a measure of spread for a covariance matrix is the determinant.

2. Compute the graph's shortest path between each pair of points according to the weight of the edge.

3. Perform step three of the Isomap algorithm.

4.4.2 Classification in the reduced space

The S-Isomap, as it provides the best category separation in the reduced space, is chosen to be used as a pre-processing step prior to application of a classification algorithm.

The S-Isomap is incorporated into a classifier as follows. First the low-dimensional manifold is estimated and the embedded mapping learned using a Generalized Regression Neural Network (GRNN) [71]. Second, a k-NN classifier in this reduced space is trained to categorize new formulations. k-NN algorithm has been shown to work well as a simple post dimensionality reduction classifier [66; 73; 74]. Finally, classification is performed on the inputs misclassified by the k-NN classifier based on the Euclidean distance of the input from each category set's centroid (a k-means [75] algorithm is used to compute the coordinates of the centroids). In the rest of the chapter we use the name 'S-Isomap based model' for the above-described approach.

Estimation of manifolds of dimension 2 to 6 were experimented with, for the first step of the above described routine, and a manifold of dimension 3 was chosen (as it allowed for the best performance of the resulting overall S-Isomap based classification model). A comparison of prediction results of the S-Isomap based model with classifiers such as ANN [76], SVM [77], Decision Tree [78] and k-NN in the same data set is performed (the ANN, SVM, Decision Tree and k-NN models are applied to full dimensional space).

4.5 Results and Discussion

4.5.1 Visualisation

Figures 4.0 and 4.1 show the results of application[5] of the five dimensionality reduction algorithms. Results are presented in the reduced space. Any correlations of data points that can be expressed as continuous functions within the new system of coordinates reveal potential underlying physical properties that determine the

[5]The Matlab toolbox provided by [69] was used for all techniques except S-Isomap. The S-Isomap Matlab code used was provided by [70].

behaviour of the system. Supervised dimensionality reduction techniques, such as LDA and the S-Isomap, make use of class labels to guide the projection of points onto the reduced space and are expected to provide better class separation as opposed to unsupervised techniques, where the class information is only used after the projection has taken place, namely, to indicate the classes that points belong to in the reduced space. The hope is that, as a result of dimensionality reduction, cluster formation according to class in the reduced space can be identified.

In addition to visual inspection of the results, we use the *global silhouette index* [79] to measure class/cluster compactness and separation achieved by each of the six techniques. The *silhouette index* was chosen as it combines ideas of both cohesion and separation. The *global silhouette index* is computed as follows [80]. Let $\boldsymbol{x}_i$, $i = 1, ..., N$ (N is the total number of points), represent a point in the reduced space. First, calculate the average distance, a_i, between $\boldsymbol{x}_i$ and the rest of the points in the reduced space that are of the same class as $\boldsymbol{x}_i$. Following that, compute the average distance, b_i, between $\boldsymbol{x}_i$ and the points from its closest class. The *silhouette width*, s_i, of $\boldsymbol{x}_i$ is then defined as

$$s_i = \frac{b_i - a_i}{max(b_i, a_i)}. \tag{4.6}$$

Hence, $-1 \leq s_i \leq 1$. A value of s_i close to 1 would indicate that $\boldsymbol{x}_i$ is well clustered according to class. A value of s_i close to -1 would indicate that $\boldsymbol{x}_i$ is not. A value of s_i close to 0 would imply that $\boldsymbol{x}_i$ lies close to the border between two classes/clusters. We now proceed to compute, S_j, the silhouette of the class/cluster C_j, $j = 1, ..., m$ (m is the number of classes)

$$S_j = \frac{1}{n_j} \sum_{i:\boldsymbol{x}_i \in C_j} s_i, \tag{4.7}$$

where n_j is the cardinality of C_j. The *global silhouette index*, S_m, is then

$$S_m = \frac{1}{m} \sum_{j=1}^{m} S_j. \tag{4.8}$$

As $-1 \leq S_m \leq 1$, the closer the value of S_m is to 1, the better the corresponding clustering is.

As can be seen, for most methods of visualisation there is no clear class separa-

tion, apart from the results of the S-Isomap algorithm: there is much better class separation in the 3D coordinates for the larger data set (Figure 4.0) and clean class separation in the 2D coordinates for the second (Figure 4.1), smaller data set. As can be seen from Table 4.1, the *global silhouette index* of the S-Isomap algorithm is the highest, non-negative and comfortably above zero, which indicates that the majority of the points in the reduced space are well clustered according to class. Thus, the dimensionality reduction and visualisation methods suggest the presence of correlations and hence the possibility of property prediction on the basis of the trained model.

S-Isomap	Isomap	LLE	PCA	LDA
0.4408	-0.0409	-0.0825	-0.028	0.164
0.445	-0.2962	-0.2182	-0.2796	-0.182

Table 4.1: *Global silhouette index* computed following the application of the five dimensionality reduction techniques to the first and second datasets (for the first dataset, the high viscosity liquid category points, as there are only four, were excluded from the computations). For the first dataset the dimensionality was reduced to 3 (for all techniques). For the second dataset the dimensionality was reduced to 2 (for all techniques).

4.5.2 Classification

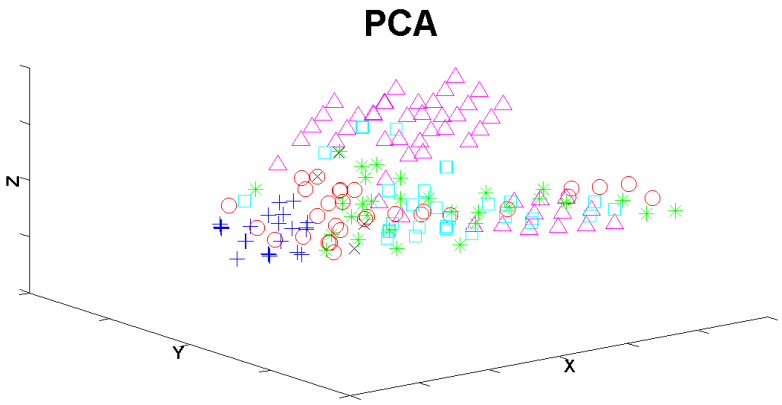
Classification was performed (on the first dataset only) using the S-Isomap based model, SVM, ANN, Decision Tree and k-NN classification algorithms. Ten times ten-fold cross validation was used in the models performance assessment. That is, for each model, the original data set was randomly divided into ten equal-sized subsets while keeping the proportion of the instances in different categories. Then, in each fold, one subset was used as a testing set and the remaining ones were used as a training set. The average result of the ten folds was recorded. This procedure was repeated ten times. In more detail, the classifiers were applied as follows. For the S-Isomap based model, for each fold during cross-validation:

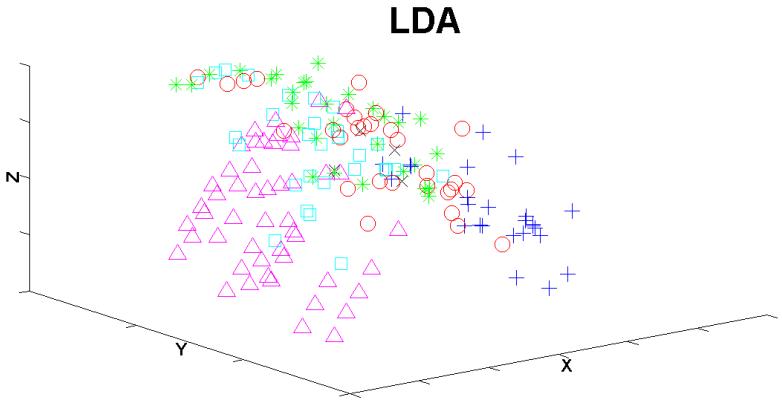
1. Using a k-means algorithm, calculate the coordinates of the centroid of each category set within the training set.

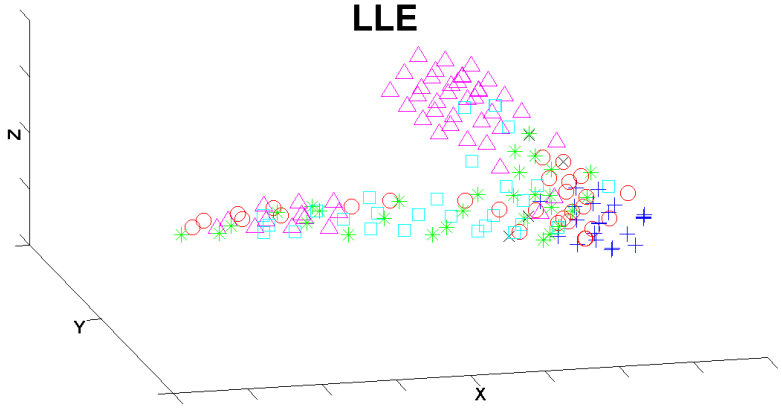
2. Map the data from the training set into a reduced feature space using the S-Isomap.

3. Construct a Generalized Regression Network to approximate the mapping.

4. Map the test set using the Generalized Regression Network and then predict the category of each point within the test set using a k-NN classification


PCA
Z
Y
X


LDA
Z
Y
X


LLE
Z
Y
X

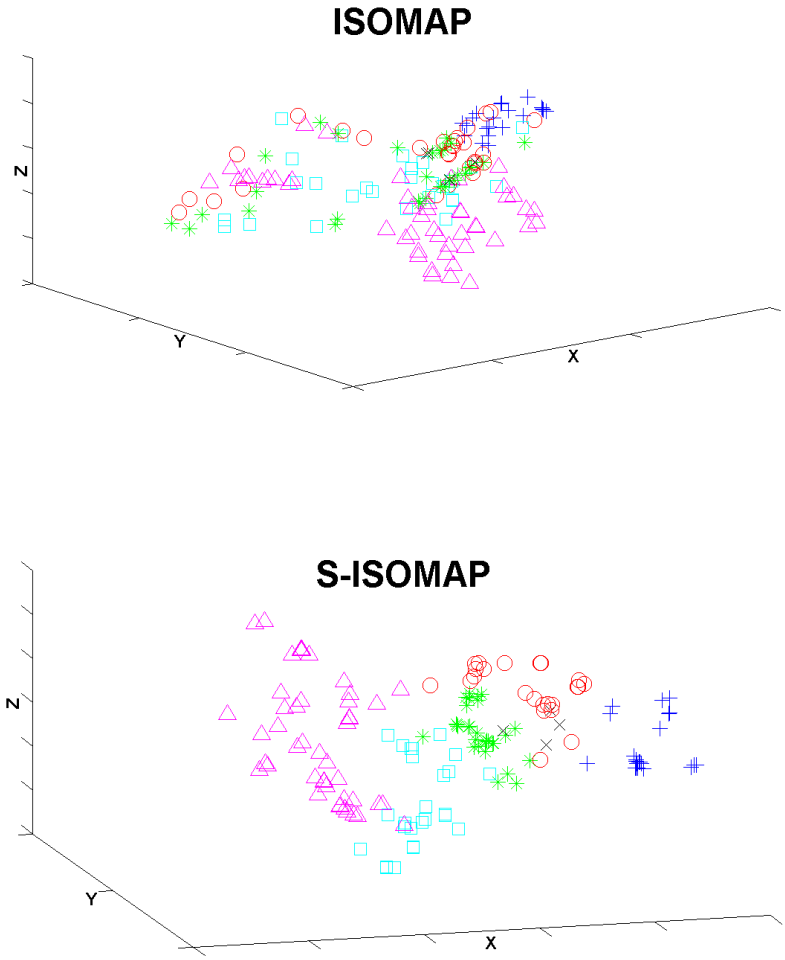
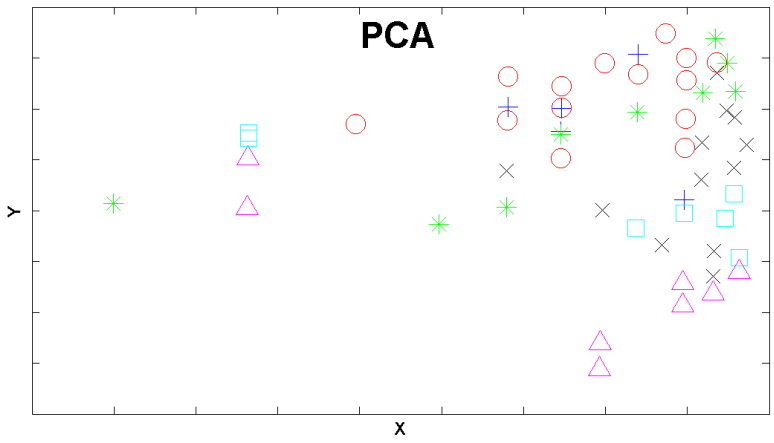
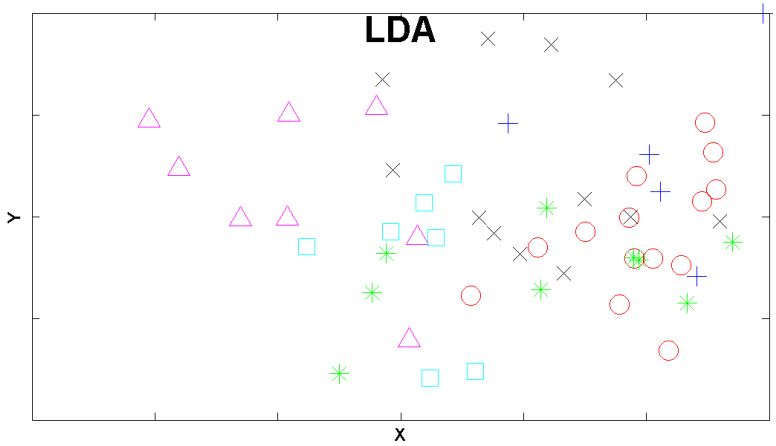
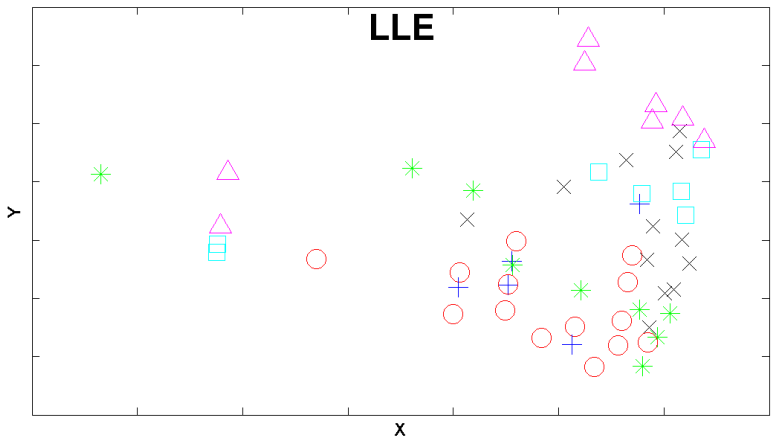


Figure 4.0: Results of applying PCA, LDA, LLE, the Isomap and the S-Isomap dimensionality reduction techniques to the first dataset. Output formulation categories are denoted as follows: strong gel '+', medium gel '$\circ$', weak gel '$*$', high viscosity liquid '$\times$', medium viscosity liquid '$\square$', low viscosity liquid '$\triangle$'. The k value (number of neighbours) for the Isomap, LLE and the S-Isomap was chosen through experimentation. Values between 5 and 15 where tested and $k = 9$ chosen.


PCA
Y
X


LDA
Y
X


LLE
Y
X

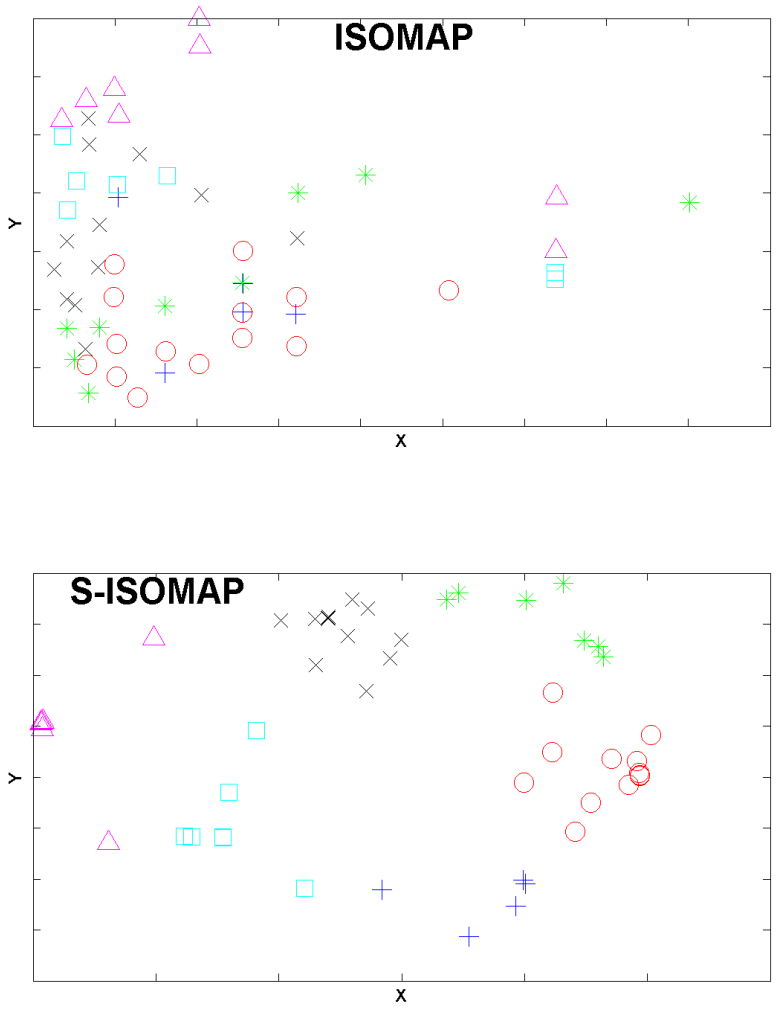


Figure 4.1: Results of applying PCA, LDA, LLE, the Isomap and the S-Isomap dimensionality reduction techniques to the second dataset. Output formulation categories are denoted as follows: strong gel '+', medium gel '$\circ$', weak gel '$*$', high viscosity liquid '$\times$', medium viscosity liquid '$\square$', low viscosity liquid '$\triangle$'. The k value (number of neighbours) for the Isomap, LLE and the S-Isomap was chosen through experimentation. Values between 5 and 15 where tested and $k = 9$ chosen.

algorithm.

5. For each misclassified point, find the category whose centroid it is closest to (using Euclidean distance) and classify it accordingly.

For the SVM, ANN, Decision Tree and k-NN classifiers, for each fold during cross-validation:

1. Using the k-means algorithm, calculate the coordinates of the centroid for each category set within the training set.

2. Train the classifier using the training set.

3. Apply the classifier to predict the category of each point within the test set.

4. For each misclassified point, find the category whose centroid it is closest to (using Euclidean distance) and classify it accordingly.

The viscosity categories were mapped onto an integer scale, prior to the application of the classification models. The parameters for all of the methods were determined empirically through experimentation. For the S-Isomap based model, different values of α between 0.1 and 1.0 and k between 5 and 40 were tested ($\alpha = 0.65$ and $k = 38$ were chosen). When applying k-NN algorithm, values of k from 5 to 40 were tested ($k = 35$ was chosen); for the SVM, two nonlinear kernels: a radial basis function and a polynomial of degrees two to six were tested (a polynomial of degree three was chosen, as it allowed for best generalisation performance for the SVM method). For the ANN, a feed-forward network with backpropagation was used (the Levenberg-Marquardt algorithm was applied). Networks with the number of nodes from three to 10 within the hidden layer were tested (a network consisting of five neurons was chosen). To prevent overfitting, an early stopping technique was used during training. Namely, when the validation error increased for a specified number of iterations (the number[6] was set to 6 in our case), the training stopped and the weights and biases at the minimum of the validation error were recorded. For Decision Tree, different values of n between five and 15, where n is the number such that impure nodes must have n or more observations to be split, were tested ($n = 6$ was chosen) at a tree fitting stage. The best level of pruning was estimated through crossvalidation (where the best level is the one that produced the smallest tree that is within one standard error of the minimum-cost subtree).

[6]This is the default setting within neural network training tool as part of the Mathworks' Neural Network Toolbox.

The mean performance, i.e. the average correct rates of the ten times ten-fold cross validation along with the corresponding standard deviations of the S-Isomap based model, k-NN, ANN, SVM and Decision Tree classifiers on the data are shown in Table 4.2. As can be seen from Table 4.2, the S-Isomap based model, SVM, ANN and

S-Isomap based model	k-NN	SVM	ANN	Decision Tree
71.64% (1.70)	59.15%(1.32)	69.61%(1.72)	71.22%(2.32)	67.32%(1.7)

Table 4.2: Mean performance and corresponding standard deviation results of the S-Isomap based classifier, SVM, ANN, k-NN and Decision Tree classifiers.

Decision Tree classifiers produced similar results (with the S-Isomap based model being, marginally, more accurate). The result produced by the S-Isomap based model though should be considered as more promising. When the S-Isomap is used for classification, the explicit mapping function from the original data space to the reduced feature space is learned by the GRNN. The quality of generalization of the classification algorithm that incorporates the S-Isomap depends on how well the GRNN can approximate the mapping. A GRNN will not approximate well (or fail) when the training data are sparse or when the algorithm is presented with a test input that is outside the range of the inputs that the GRNN was trained on. It is hoped that a substantial improvement in classification rate can be achieved once a more efficient approach is found to transverse between the input space and the reduced feature space for the out-of-sample data points. One, possibly more efficient approach, could be to purposefully extend the S-Isomap algorithm to enable it to embed into the reduced space both the labelled and the unlabelled data (test inputs) simultaneously, thus avoiding the need for approximating the mapping altogether. To this end one could explore the idea presented in [81], where the authors employ label propagation to assign virtual labels to the unlabelled data prior to the application of dimensionality reduction. Although further optimizing of the parameters used within SVM and ANN classifiers may improve their prediction accuracy further, the gains are anticipated to be marginal. It is useful, in terms of further assessment of the classifiers' performance, to know the typical (for the data investigated) acceptable performance level. However, at present, we do not have such information.

4.6 Conclusions

In this chapter we have described the application of dimensionality reduction to two datasets of formulations for visualisation and as a pre-processing step prior to clas-

sification. In total, five dimensionality reduction techniques were employed. As the datasets also included class labels (description of viscosity as one of six categories), supervised dimensionality reduction algorithms were included among the six techniques. The algorithms chosen to perform supervised dimensionality reduction were the S-Isomap and LDA. The S-Isomap algorithm, as it provided the best class separation in the reduced space, was used as a pre-processing step prior to classification. As the S-Isomap does not have an internal model, the mapping was learned using a Generalized Regression Network. The lower dimensional coordinates were then used as inputs to train a k-NN classifier for prediction of out-of-sample formulations.

For visualisation, the S-Isomap, compared to the rest of the dimensionality reduction techniques experimented with, produced the best results. Using the S-Isomap as a pre-processing step in classification of out-of-sample formulations a classification rate was achieved that is on a par with classification rates achieved by SVM and ANN classifiers (see Table 4.2).

The S-Isomap based model, could also be used in reverse, as a GRNN can be built to map points from the reduced space into the original full dimensional space. This would allow for a rapid generation of a number of formulations corresponding to the same category of the system property in question. The inefficiency of a GRNN at mapping sparse data, however, would still be an issue to be addressed.

Both, the application of the S-Isomap algorithm for visualisation and as a pre-processing step prior to classification are yet to be tested on a variety and a large number of formulation data sets. Therefore, no generalisations are made at this stage.

Chapter 5

Concept development

5.1 Brief summary of the chapter

This chapter, as opposed to previous chapters, provides a different prospective on chemical product design and optimisation. Where as in chapters 2 and 3 we discussed target value product development where all of the input variables are uniquely identified, in this chapter we contribute to the design of product development methodology that relaxes the input variables 'uniqueness' constraint. In addition, in this chapter we, specifically, look to investigate the incorporation of bio-feedstocks in consumer product design.

Our current approach to formulated product design is based on heuristic knowledge of formulators that allows selecting individual compounds from a library of available materials with known properties. We speculate that most of the compounds (or functions) that make-up the product to be designed can potentially be obtained from a single or very few bio-sources. In this case, it may be possible to design a sequence of transformations required to transform feedstocks into products with desired properties, analogous to a metabolic pathway of a complex organism. We conceptualise some novel approaches to processing bio-feedstocks with the aim of bypassing the step of a fixed library of ingredients. Two approaches are brought forward: one making use of knowledge-based expert systems and the other making use of applications of metabolic engineering and dynamic combinatorial chemistry.

5.2 Introduction

Global uncertainty over prices of petrochemical feedstocks and the desire to significantly reduce the levels of anthropogenic generation of CO_2 are the two main drivers behind current rapid development of a replacement supply chain for platform molecules of the chemistry using industries [15; 16]. What platform molecules produced by new bio-refining technologies [17; 18; 19] would form the basis of the new supply chain is still an issue of a significant debate. This, however, significantly affects downstream technologies, which depend on the catalogue of available molecules to develop task-specific products, for example in formulations.

At present the use of bio-feedstocks in product design is relatively limited, due to a small range of molecules available on the market, primarily natural oils, flavour and fragrance substances, nutraceuticals, bio-pharmaceuticals. Very few bio-derived solvents, surfactants, monomers are available at present. However, this range is expected to be rapidly expanded, offering new opportunities for product design. The emerging question is whether our existing methods of product design in formulations and other chemistry-using industries are appropriate for the new developing supply chain based on sustainable renewable feedstocks?

Our current approach to formulations design is based on heuristic knowledge of formulators that allow one to select individual compounds from a library of available materials with known properties, i.e. rheology modifiers, structure-forming agents, colour and fragrance substances, bio-actives etc. [20]. The new bio-feedstocks based supply chain will replace some of the usually applied ingredients or offer new ingredients with different functionalities. This is represented by the lower path in Figure 5.1, from bio-feedstocks to the final products. The main difference with the current petrochemical-derived supply chain of ingredients to formulations may be in a broader specification of properties of the ingredients due to variability in bio-feedstocks.

We speculate that most of the compounds (or functions) that make-up the final product can potentially be obtained from a single, or very few, bio-sources. In this case, there could be an alternative path from feedstocks to products, analogous to a metabolic pathway of a complex organism. This is represented by the top path in Figure 5.1.

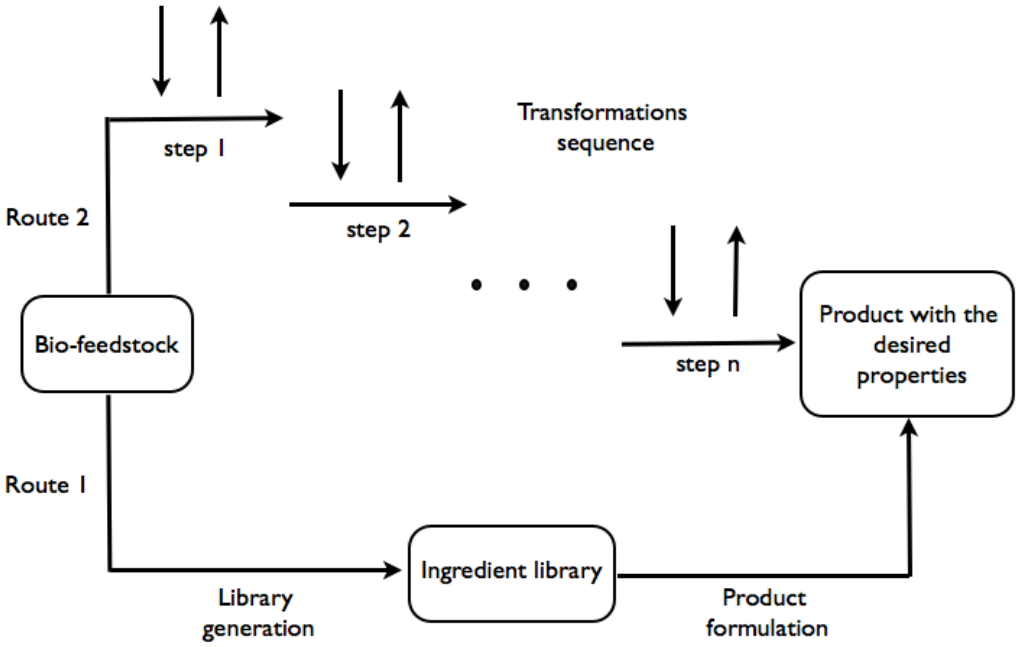


Figure 5.1: Schematic representation of both current and proposed approaches to processing bio-feedstocks in product design.

In the last four decades there has been a growing interest in automatic creation of retrosynthetic designs via employment of computer programs (written by expert chemists) – expert systems. The limitation of these systems used to be the fact that the rules[1] they were based on were programmed by hand and, consequently, it was very difficult to keep updating the rules as the chemistry developed [88]. The rapid increase in computing power together with the arrival of databases, however, have changed that. Millions of chemical reactions performed, and compounds synthesized to date have been systematically recorded and incorporated into a variety of product and reaction databases [82]. The availability of such databases has prompted the development of algorithms and software tools [83; 84; 85; 86; 87; 88; 89], collectively known as expert systems, to help intelligently explore the gathered information. Some expert systems are designed to predict major products of a reaction given a combination of starting materials and reagents [84; 85; 86; 87], while others are designed to predict, using retrosynthetic analysis, possible starting materials given a target product [88; 89]. Different but nonetheless potentially viable approaches

[1]The rules themselves are not discussed in this thesis. For examples and detailed description of some of the rules see [83], for instance.

to chemical product synthesis are also being developed within combinatorial chemistry. In combinatorial chemistry combinatorial libraries are built (through forward synthesis) and screened for products of interest using a deconvolution approach, for instance. With deconvolution, a series of compound mixtures are synthesized combinatorially, with each iteration correcting some specific structural feature. Each mixture is then examined and the most active combination is investigated further. Further rounds systematically correct other structural features until a manageable number of discrete structures can be synthesized and screened. Deconvolution, for instance, can be used to optimize the most active peptide sequence from a large number of possibilities [99].

Alternatively, other approaches to product synthesis are being developed, that do not rely on the information stored in product and reaction databases. These approaches build on advances made in the fields of metabolic engineering [92; 93] and dynamic combinatorial chemistry [94; 95].

The main element of expert systems based approaches is existing chemical knowledge of a large number of compounds and reactions. Recorded in the form of on-line databases, this knowledge is in a format that allows interrogation and rule generation to be performed using expert systems. We speculate that existing chemical knowledge will now include the necessary information for expert systems to, using retrosynthetic analysis, generate (or go some way towards) synthetic routes connecting bio-feedstocks (as starting material) with a number of existing products. Also, as more bio-derived molecules with a variety of different functionalities, are added to the existing supply chain of ingredients, and the corresponding information (the molecules and the starting materials that were used in creating them) is transferred into the existing chemical knowledge, it becomes increasingly possible that synthetic routes connecting bio-feedstocks with *new products* having desired properties, will be found using approaches incorporating expert systems and other chemical knowledge using approaches, such as combinatorial chemistry based techniques, for instance.

Also, new approaches to processing bio-feedstocks in designing products with specific properties through advances made in metabolic engineering and dynamic combinatorial chemistry are envisioned. Within metabolic engineering metabolic pathways are assembled and optimized (by tuning the activity of the intermediate reaction steps) for the production of molecules with desired properties. Note that there

are parallels with approaches in synthetic biology whereby metabolic pathways are assembled *in toto* by combining genes isolated from a variety of sources. While pathway construction is similar to approaches by metabolic engineering, it's optimization is different. In synthetic biology the search for an optimal pathway configuration is performed via the use of gene-combinatorial methods whereby large numbers of pathways, comprising a number of combinations of genes from different sources, and their mutants, are evaluated [96]. In dynamic combinatorial chemistry dynamic combinatorial materials are designed as artificial chemical systems that display modulation of functional properties in response to the application of external parameters [95]. These approaches may be viewed as evolution based, as the final product could be arrived at through evolution of the systems involved. For instance, directed evolution is employed in optimizing enzymes and biosynthetic pathways [97] involved in the synthesis of commercial products [98]. Typically, the directed evolution cycle of an enzyme involves: diversification of the parent gene, via a chosen method of random mutagenesis and/or in vitro gene recombination; mutant enzymes production, using the library of mutant genes; and identification of improved enzymes, whose genes will be used as parents in the next cycle, through a high-throughput screening or selection method [100]. In dynamic combinatorial chemistry equilibrating libraries of building blocks are generated, under reversible conditions, and evolve based on a selection process [101]. The components of a dynamic combinatorial chemistry experiment are loosely related to components of a system undergoing Darwinian evolution in that a population of individuals (the library), reproduction (based on reversible reactions) and selection (based on binding interactions) are present [102]. However, introduction of new diversity into the library of building blocks and iteration of the process are still an issue [103].

5.3 Methods

5.3.1 Expert systems based approach

Two variants of one methodology making use of existing chemical knowledge and expert systems are proposed, one aiming for products with known composition and the second targeting known functional properties, but with an unknown composition.

First variant

Given that the target product composition is known, use expert systems and existing chemical knowledge to find an optimal sequence of transformations to perform

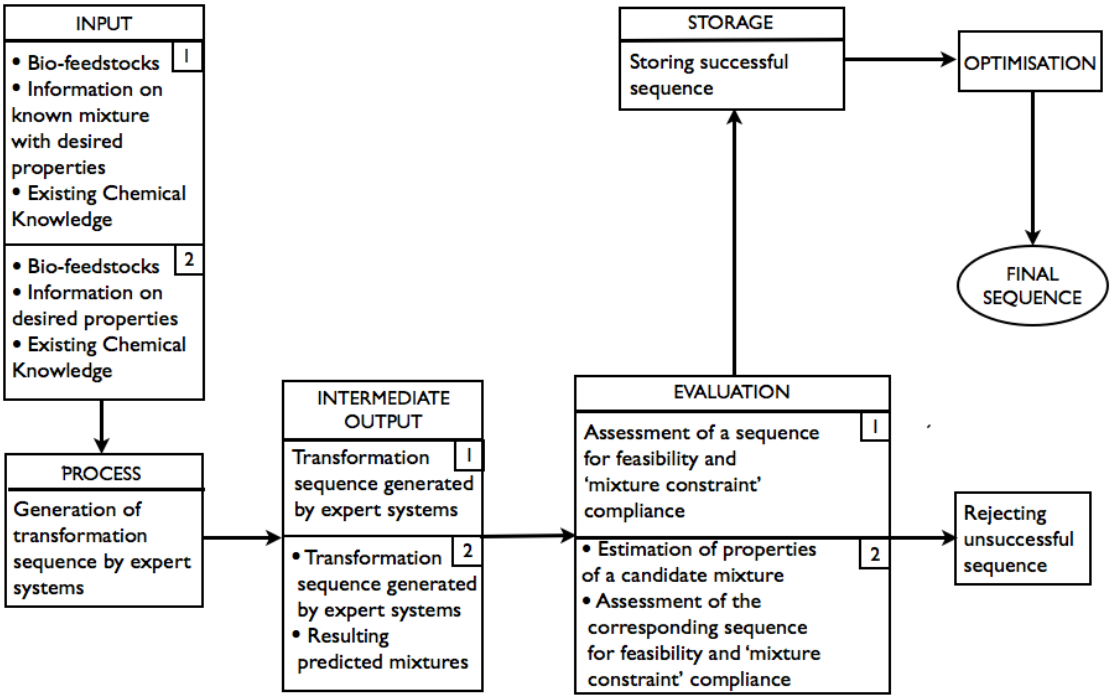


Figure 5.2: Schematic representation of the expert systems based approach: route 1 - with known product composition; route 2 - product composition is unknown.

on bio-feedstocks to arrive at the desired product.

As shown in Figure 5.2 (route 1), bio-feedstocks, the information on composition of the target product/mixture and existing chemical knowledge are **INPUT**. The generation of transformation sequences is the **PROCESS**. At this stage, expert systems are used to perform retrosynthetic analysis. **INTERMEDIATE OUTPUTS** are all sequences 'deemed' possible (but not necessarily feasible) by the expert system. At the **EVALUATION** stage, the sequences that are returned as possible are assessed by the expert systems for feasibility. As the transformations are to be performed on a mixture of bio-feedstocks, it is important to ensure that, for any reaction in a sequence, the properties of the components that do not take part in the reaction remain unaffected [104] (referred to as 'mixture constraint' in the diagram). Thus, each feasible sequence is also required to satisfy the 'mixture constraint'. It is possible that, after all of the generated sequences have been evaluated, there would be more than one acceptable transformation sequence in **STORAGE**, in which case, there would be opportunities for **OPTIMIZATION**. To date, to the best of our knowledge, expert systems based on retrosynthetic analysis are yet to be employed in the synthesis of commercial products. However, the capabilities

of such expert systems have been validated. For instance, in [88], Route Designer expert system was able to find a synthetic transformation sequence for Zatosetron, a potent, selective and long acting 5HT[2] receptor antagonist used in the treatment of nausea and emesis associated with oncolytic drugs. First, by rule extraction, using the MOS reaction database [91] from Accelrys and the Beilstein Crossfire reaction database from Elsevier, was performed. The program was then presented with the target and a database of 120 thousand of starting materials. The search for possible synthetic sequences followed. The assessment of feasibility of suggested sequences is inbuilt in the algorithm and was performed at a transformation level as the sequences were assembled. A number of ranked possible, feasible sequences were then presented as the output.

Second variant

Given only the desired properties of the product are known, use expert systems and existing chemical knowledge to perform a series of transformations on bio-feedstocks to arrive at a product with the desired set of properties.

As shown in Figure 5.2 (route 2), a mixture of bio-feedstocks and the information on the product's properties are **INPUT**. Generation of possible transformation sequences is, again, the **PROCESS** and involves the use of expert systems to perform forward synthesis. **INTERMEDIATE OUTPUTS** are all sequences and the corresponding predicted resulting products 'deemed' possible (but not necessarily feasible) by the expert system. At the **EVALUATION** stage, the sequences that are returned as possible are assessed by expert systems for feasibility. The properties of predicted resulting products of feasible sequences are estimated. The sequence is accepted if the estimates of the properties are within the tolerance required and the 'mixture constraint' is not violated. Again, it is possible that there would be more than one acceptable transformation sequence, in which case, there would be an opportunity for optimization.

At present, within the context of the above mentioned approach, expert systems are able to assist with the generation and evaluation (in terms of feasibility only) of the candidate products. In [84], for instance, examples are given of an expert system, with reaction predicting capabilities, being used to generate products in two ways:

[2]5HT stands for 5-hydroxytryptamine receptors, a group of receptors found in the central and peripheral nervous systems [90].

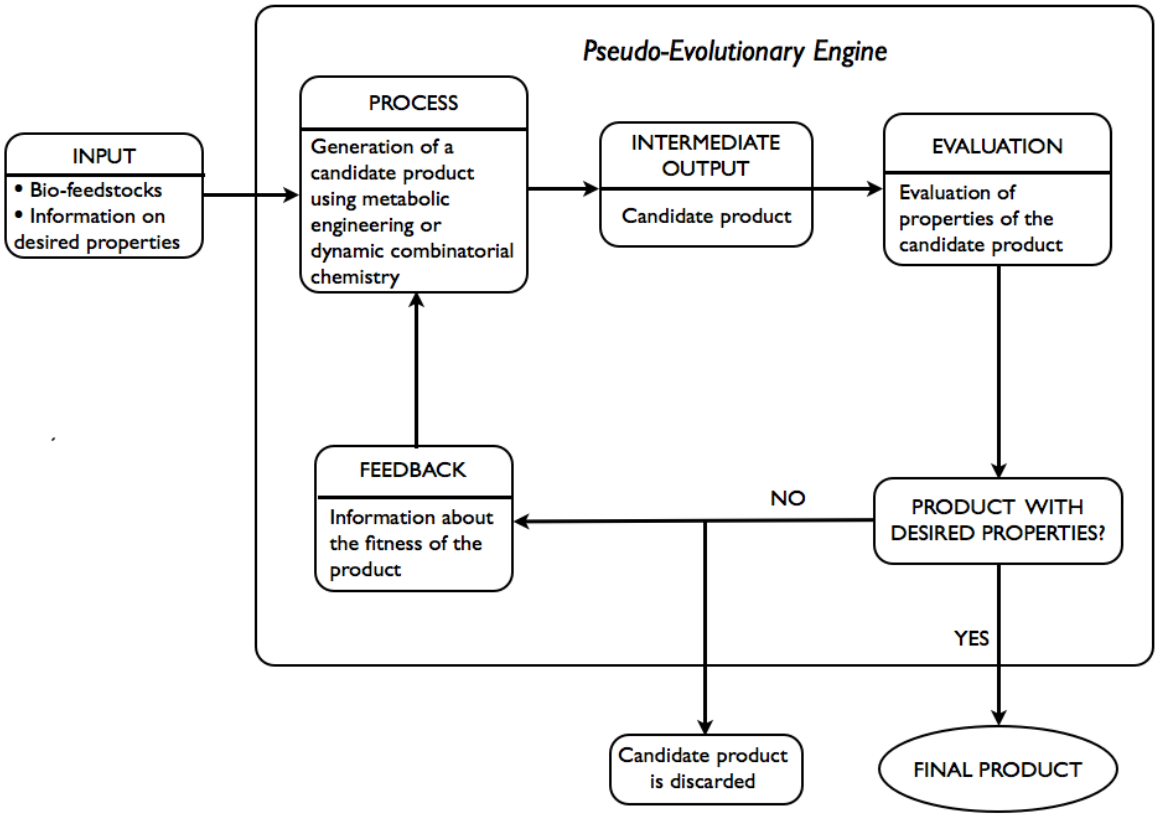


Figure 5.3: Schematic representation of evolution-based approach.

generating products systematically, given a number of predetermined starting materials; and generating products that are similar to a given target product using the precursors of the target product as starting materials. Also, the design of the expert system ensures that it only predicts synthetically feasible products. The scheme in full, as in Figure 5.2 (route 2), has not been attempted, to the best of our knowledge.

As further details of the expert systems based approach are formalised, it would be of interest, in our view, to investigate the possibility of casting the associated optimisation problem into an optimal control problem framework.

5.3.2 An evolution-based approach

The second approach involves the use of approaches developed in the fields of metabolic engineering and dynamic combinatorial chemistry. Given only the desired properties of the product are known, a pseudo evolutionary engine is used, as illustrated in Figure 5.3, to arrive at a product with the desired properties. **INPUT**

includes bio-feedstocks and information on desired properties of a product. At the **PROCESS** stage an **INTERMEDIATE OUTPUT**-candidate product is generated via the application of metabolic engineering or dynamic combinatorial chemistry. At the **EVALUATION** stage an assessment of candidate product, in terms of whether the estimates of desired properties are within required tolerance or not, is performed. If the candidate product does not meet the requirements, it is discarded. Adjustments to the generation step are made and a new candidate product is generated. The process is repeated until desired properties are within the tolerance required.

The majority of components of the above mentioned approach are in existence. Within metabolic engineering, a number of strains are often designed (**PROCESS**, **INTERMEDIATE OUTPUT**) and evaluated in terms of percentage yield of the desired product (**EVALUATION**). Following the evaluation, adjustments are made and often involve deletion of unnecessary or optimization of necessary genes or a combination of thereof. Within dynamic combinatorial chemistry, dynamic combinatorial libraries[3] (DCLs) are constructed (**PROCESS**) and evaluated in terms of functional modularity in response to an external stimuli (**INTERMEDIATE OUTPUT**, **EVALUATION**). Following the evaluation, adjustments to the construction of DCLs are made and often involve: replacement of some of the constituents within DCLs; alteration of the reversible chemistry used; a change in the external stimuli applied. For instance, in [105] authors report the use of metabolic engineering in designing a synthetic pathway in *E. coli* for the production of isopropanol [106], a secondary alcohol that is used, among other ways, in pharmaceutical applications. The successful strain was developed through expression of a variety of combinations of genes from a selected list of known strains. In [107] the authors give an account of the use of metabolic engineering and enzyme engineering in the development of *E. coli* strains for the production of biomass-derived plastics, polylactic acid, and its copolymers. The techniques applied in optimizing metabolic pathways of the organism were deletion of unnecessary genes and optimization of the expression of necessary genes based on in-silico genome-scale flux analysis combined with a rational approach.

In [108; 109; 110; 111] the authors make use of dynamic combinatorial chemistry in the identification of enzyme-inhibitors. Commonly, the steps involved include

[3]In dynamic combinatorial chemistry, a DCL is a collection of reversibly interconverting building blocks that, through reversible reaction (and under thermodynamic control), form new molecules [101].

generation of DCLs under thermodynamic control using a predetermined type of reversible chemistry and assessment of interactions between constituents of the libraries and the target through 'measuring' of the change in the composition of a DCL upon introduction of a target.

5.4 Merits and drawbacks of the proposed approaches

The first variant of the expert systems based approach is, perhaps, the easiest to implement. As the target product is known from the start, this variant does not involve property estimation, which can be difficult to do analytically and costly (if carried out in the lab). Its main drawback is the fact that a transformation sequence, connecting given starting materials with a target product, might simply not exist, as some or all of the required chemistry may not have been carried out yet. The second variant of the expert systems based approach, however, although also dependant on a large variety of chemistry to have been done, is not constrained by the necessity of finding a transformation sequence to a given target product, but rather by a set of properties that the target product should have, which somewhat liberates the search. In fact, the final output of this approach may, intriguingly, be a product that has not been considered before. The main foreseen difficulty with this variant is the costs associated with the necessity for property estimation, either analytically or through an experiment, for each candidate product.

The main advantage of the evolution-based approach, as opposed to the expert systems approach is the potential for discovery of novel products with desired properties and the potential to discover new knowledge. The biomimetic approach of evolution-based process development requires the implementation of generic principles of evolutionary development, which will necessarily sample a very large space of potential process variants. This methodology depends on the ability to sample the outcomes of each evolutionary step and to make adequate decisions, both, about the new, yet unknown phenomena that took place and which could potentially be exploited, as well as about the following steps in the process evolution. As in natural evolution, the approach is not blind, but follows some generic rules. The envisioned evolutionary approach would, at the very basic level, involve an 'adjustment' (mutation) step, applied iteratively, to evolve a product generating process. However, it is not unreasonable to think of the possibility of evolving a population of product generating processes, in which case selection and crossover steps would come into play. To allow the approach to converge on the optimal (near optimal) process (or

population of processes) within the allocated amount of resources and/or time, adequate selection, crossover and mutation operators would need to be designed.

The evolution-based approach has the potential to not only discover novel products with desired properties, but, intriguingly, products with additional, perhaps unexpected, functions/properties. These (additional functions/properties), of course, can be undesirable and the candidate product discarded, in the context of the product sought. However, the *new knowledge*, thus acquired, may benefit the design of new products and, hence, should be retained. In addition to the potential to facilitate development of other products with different functionality, the data collected using the evolution-based approach could be utilised to build physical/empirical models of the underlying physical processes involved in product generation (or help improve the existing methodology).

5.5 Conclusions

In this chapter designing new ways of processing bio-feedstocks in consumer product design was discussed. An attempt was made to conceptualise some novel approaches to processing bio-feedstocks with the aim of bypassing the step of a fixed library of ingredients. Two approaches were brought forward and discussed: one making use of expert systems and the other, evolution-based approach, making use of advances made in the fields of metabolic engineering and dynamic combinatorial chemistry. The two main components of both approaches are: generation of a number of candidate transformation sequences/process variants and properties estimation of the candidate products (second variant of the expert systems based approach and the evolution based approach). Both (components) present challenges. In silico generation of candidate products/transformation sequences using expert systems, given sufficient information is contained within existing chemical knowledge, is very time and material efficient, however, properties estimation of the candidate products (second variant of the expert systems based approach) is likely to be material-intensive and time-consuming. By contrast, the generation of candidate products using an evolution-based approach involves the design and set up of experiments, which may require a substantial investment of time and resources. However, this initial investment would pay off at the properties estimation stage. It is possible that the problem of identifying the transformations needed in processing bio-feedstocks could be solved through the use of both approaches, as some transformations may only be done via an expert systems based or evolution-based approach.

Chapter 6

Discussion and Outlook

The work presented in this thesis is devoted to the development of new ideas and techniques for the acceleration and automation of processes involved in the design of formulated or otherwise synthesized chemical products with predefined properties. In particular, techniques were presented that address the shortcomings of the existing methods and provided a bird's-eye view over the new possible directions for chemical product development necessitated by the integration of bio-feedstocks into the existing supply chain. In the following, we summarize the results covered by this work. For each area, we discuss what has been achieved and outline directions for future research.

Sequential Multi-Target Optimization of Expensive-to-Evaluate Functions Multi-target optimization is an integral part in the development of chemical products/processes with predefined properties. There is a need for techniques capable of searching for near optimum solutions in the most efficient (time, material resource) manner. Sequential optimization algorithms that incorporate DOE are particularly suited for this task. In chapter 2 we introduced a novel algorithm for multi-target optimization of expensive-to-evaluate functions that is equipped to sequentially choose the next set of experimental conditions that are predicted to be optimal both in terms of providing information about the underlying approximated physical processes and in terms of the proximity of predicted values of the underlying processes to the targets. To achieve this, the algorithm combines application of Gaussian Processes and Mutual Information. It also incorporates a Genetic Algorithm, to allow for an increased resolution when searching the decision space. To illustrate the potential use of the approach, it was applied to simulate the optimization of target values of fictitious physical processes. Constrained and unconstrained

optimization, using the proposed algorithm, was illustrated. The algorithm was compared against the surrogate based on-line Evolutionary Algorithm specifically designed for optimization of expensive-to-evaluate functions. Results indicate that, using the hypervolume indicator as performance criteria, the proposed approach compares favourably against the surrogate based on-line Evolutionary Algorithm on tasks involving small budget of evaluations. Although the computational complexity of the proposed algorithm is high, it is not foreseen to be a hindrance for the type of applications it is designed for. In this chapter we also extended the algorithm to enable it to, using user interaction, tackle situations where some of the constraints on the decision space are unknown. This is achieved via incorporation of a binary classification model that predicts a category (feasible/infeasible) for each point within the decision space prior to the optimization step.

The MOAL algorithm was also applied for multi-target optimization of a copolymerization reaction where the requirement was to, in as few experiments as possible, find reaction conditions that: provide maximum conversion; produce, on average, polymer particles of 100 nm in diameter. Using the algorithm, we were able to find an optimal solution within a modest number of expensive evaluations. Also, analysis of the optimal solutions obtained via extended optimization has led to discovery of a variety of different modes of optimal solutions. We were also able to detect an input-output variable dependency. Directions for future work include:

- Development of a systematic approach for higher level model (GP covariance function) selection, for situations where little or no prior knowledge about the underlying process is available.
- Development of an approach to incorporate the classification model within the automated feedback loop of the algorithm.

Application of dimensionality reduction In chapter 4 we described the application of dimensionality reduction to two datasets of formulations for visualisation and as a pre-processing step prior to classification. In total, five dimensionality reduction techniques were employed. As the datasets also included class labels (description of viscosity as one of six categories), supervised dimensionality reduction algorithms were included among the five techniques. The algorithms chosen to perform supervised dimensionality reduction were the S-Isomap and LDA. The S-Isomap algorithm, as it provided the best class separation in the reduced space, was used as a pre-processing step prior to classification. As the S-Isomap does not have an internal model, the mapping was learned using a Generalized Regression

Network. The lower dimensional coordinates were then used as inputs to train a k-NN classifier for prediction of out-of-sample formulations. For visualisation, the S-Isomap, compared to the rest of the dimensionality reduction techniques experimented with, produced the best results. Using the S-Isomap as a pre-processing step in classification of out-of-sample formulations a classification rate was achieved that is on a par with classification rates achieved by the SVM and ANN classifiers. The S-Isomap based model, could also be used in reverse, as a GRNN can be built to map points from the reduced space into the original full dimensional space. This would allow for a rapid generation of a number of formulations corresponding to the same category of the system property in question. The inefficiency of a GRNN at mapping sparse data, however, would be an issue to be addressed. The following are areas where future work could be carried out:

- It is hoped that a substantial improvement in the classification rate of the presented algorithm can be achieved once a more efficient approach is found to transverse between the input space and the reduced feature space for the out-of-sample data points. One, possibly more efficient approach, could be to purposefully extend the S-Isomap algorithm to enable it to embed into the reduced space both the labelled and the unlabelled data (test inputs) simultaneously, thus avoiding the need for approximating the mapping altogether.

- Both, the application of the S-Isomap algorithm for visualisation and as a pre-processing step prior to classification need to be tested on a variety and a large number of formulation data.

Concept development The new bio-feedstocks based supply chain is replacing some of the usually applied ingredients used in chemical product design. The main difference with the current petrochemical-derived supply chain is in a broader specification of properties of the ingredients due to variability in bio-feedstocks. In this thesis we explored new possibilities for product design in view of the intuition that most of the compounds (or functions) that make-up the final product can potentially be obtained from a single, or very few, bio-sources. In chapter 5 we discussed designing new ways of processing bio-feedstocks in consumer product development. We conceptualised novel approaches to processing bio-feedstocks with the aim of bypassing the step of a fixed library of ingredients. Two approaches were brought forward and discussed: one making use of expert systems and the other, an evolution-based approach, making use of advances made in the fields of metabolic engineering and dynamic combinatorial chemistry. The two main components of both approaches are: generation of a number of candidate transformation sequence/process variants

and properties estimation of the candidate products. Both (components) present challenges. In-silico generation of candidate products/transformation sequences using expert systems, given sufficient information is contained within existing chemical knowledge, is very time and material efficient, however, properties estimation of the candidate products (second variant of the expert systems based approach) is likely to be material-intensive and time-consuming. By contrast, the generation of candidate products using an evolution-based approach involves the design and set up of experiments, which may require a substantial investment of time and resources. However, this initial investment would pay off at the properties estimation stage. The following are areas where future work is envisioned:

- It is reasonable to expect that for a lot of problems in chemical product design a large number of transformation sequences would need to be considered (through the use of expert systems), and so, to intelligently navigate the search, appropriate optimization techniques would need to be employed.
- It is possible that the problem of identifying the transformations needed in processing bio-feedstocks could be solved through the use of both of the discussed approaches, as some transformations may only be done via expert systems based ***or*** evolution-based approaches. In this situation unifying, flexible approach would be required, that would allow switching between expert systems based and evolution-based approaches when necessary.

6.1 Conclusions

To summarize, in relation to chemical product design and optimisation, in this thesis we have addressed the following points:

1. The lack of data driven techniques addressing the scenario of multi-target optimization in the presence of limited amount and/or high cost of materials.
2. The lack of real time multi-target optimization approaches where modelling, experimentation and optimization are part of a closed feed back based loop.
3. Difficulties with data classification and visualisation associated with high dimensionality of problems encountered in formulated product design.
4. The need for conceptualising new methodologies for chemical product design in view of the new developing supply chain of ingredients based on sustainable renewable feedstocks.

Points one, two and three of the above list were addressed via development of suitable algorithms discussed in chapters 2, 3 and 4. The last point was addressed via conceptual development of novel methodologies discussed in chapter 5.

6.2 Concluding remarks

We believe that the contributions presented in this thesis, in particular the MOAL algorithm presented in chapter 2, can be of great assistance in chemical product development. The MOAL algorithm can readily be incorporated into an automatic feedback loop environment wherein the optimisation is carried out without human intervention. Furthermore, the extended version of the MOAL algorithm can be an effective tool where the difficulty of the problem is increased by the presence of unknown constraints on the design space.

We believe that the contributions presented in this thesis have a high potential also beyond chemical product design. The algorithm presented in chapter 4, for instance, can be of assistance in many engineering problems described as challenging because of their high dimensionality.

Our contribution presented in chapter 5, although not at present in a format of a detailed algorithm, opens up an important discussion on new methodologies to chemical product design.

Appendix A

Matlab code for the MOAL algorithm

The code presented in Appendix A is the one employed for the simulation discussed in chapter 2 for the Ackney and the Booth functions.

The code presented in Appendix A and Appendix B is to be run in conjunction with Gaussian Process Regression and Classification Toolbox version 3.1 for Matlab by Carl Edward Rasmussen and Hannes Nickisch downloaded from http://gaussianprocess.org/gpml/code. Also, the code for the computation of the hypervolume indicator was provided by Yi Cao [47].

```
function [PtrTrain,ytrTrain] = OnlineActiveSearchRangeAckley(AA,PP,N,NI)

% Main function of the MOAL algorithm employed for the simulation
% discussed in chapter 2 for the Ackney and the Booth functions
% Input: AA — the training set
%        PP — the set of candidate solutions
%        N  — the number of points to be designed after each iteration
%             (default value is 1)
%        NI — the number of iterations for the optimisation run
% Output: ytrTrain — the set of observations collected during the
%                    optimisation run
%         PtrTrain — the set of corresponding solutions

Target = [0 log(0.001)];
PtrTrain = [];ytrTrain = [];total = length(AA(:,1));RunV = [];...
```

```
    stop = 0;count = 0;
Ptr = AA;
range = 60;
ytr(:,1) = ackleyMatrix(Ptr);
ytr(:,2) = boothMatrix(Ptr);
PtrTrain = [PtrTrain;Ptr];ytrTrain = [ytrTrain;ytr];
r1 = max(ytrTrain(:,1))-Target(1);
r2 = max(ytrTrain(:,2))-Target(2);
v = trial1(ytrTrain,Target,r1,r2);
RunV = [RunV v];
%-------------------------------
figure(10)
[X1,Y1] = meshgrid(-30:.1:30);
Z = boothMatrixContour(X1,Y1);
colormap(gray)
[C,h] = contour(X1,Y1,Z);
hold on
plot(PtrTrain(:,1),PtrTrain(:,2),'ks','markersize',5)
hold on
plot(1,3,'kx','markersize',10)
hold on
plot(1,3,'ks','markersize',10)
hold on
plot(0,0,'kx','markersize',10)
hold on
plot(0,0,'ks','markersize',10)
hold on
h0b = figure(10);
print(h0b,'-deps2','Figure0bM5.eps')
%-------------------------------
%-------------------------------
figure(11)
Z1 = ackleyMatrixContour(X1,Y1);
colormap(gray)
[C,h] = contour(X1,Y1,Z1);
hold on
plot(PtrTrain(:,1),PtrTrain(:,2),'ks','markersize',5)
hold on
plot(1,3,'kx','markersize',10)
hold on
plot(1,3,'ks','markersize',10)
hold on
plot(0,0,'kx','markersize',10)
hold on
plot(0,0,'ks','markersize',10)
hold on
```

```
h0a = figure(11);
print(h0a,'-deps2','Figure0aM5.eps')
%-----------------------------
figure(2)
xlabel('y1','FontSize',16)
ylabel('log(y2)','FontSize',16)
hold on
id = [1,1]; % use Exact
sn1 = 0.01;sn2 = 0.01;
sf1 = 1;
sf2 = 1;
cov  = {'covMaterniso',3};
ell1 = 2;
ell2 = 2;
hyp01.cov  = log([ell1 sf1]);
hyp02.cov  = log([ell2 sf2]);
mean = {@meanZero};
lik_list = {'likGauss','likLaplace','likSech2','likT'};
inf_list = {'infExactNew','infLaplace','infEP','infVB'};
for i=1:size(id,1)
    lik = lik_list{id(i,1)};                    % setup the likelihood
    hyp01.lik  = log(sn1);
    hyp02.lik  = log(sn2);
    inf = inf_list{id(i,2)};
end
%=================================================================
[Fmax1,I1,fmin1,best1] = findMaxLik(PtrTrain,ytrTrain(:,1),5,1,60);
[Fmax2,I2,fmin2,best2] = findMaxLik(PtrTrain,ytrTrain(:,2),5,1,60);
hyp1.lik  = fmin1(I1,2);
hyp2.lik  = fmin2(I2,2);
hyp1.cov  = fmin1(I1,3:end);
hyp2.cov  = fmin2(I2,3:end);
%=============================
counter1 = crossVmodel(PtrTrain,ytrTrain(:,1),hyp1);
if counter1~=0
    disp('model 1 is no good!')
end
counter2 = crossVmodel(PtrTrain,ytrTrain(:,2),hyp2);
if counter2~=0
    disp('model 2 is no good!')
end
[ymu1{1}, ys21{1}] = gpCorrected(hyp1, inf, mean, cov, lik,...
    PtrTrain, ytrTrain(:,1),PP);  % predict
[ymu2{1}, ys22{1}] = gpCorrected(hyp2, inf, mean, cov, lik,...
    PtrTrain, ytrTrain(:,2),PP);  % predict
ym1 = ymu1{1};YY1 = [min(ym1) max(ym1)];
```

```
ym2 = ymu2{1};YY2 = [min(ym2) max(ym2)];
s = [ym1 ym2];
yy = mgaFunctions(s,Target);
MAX = [max(yy(:,1)) max(yy(:,2))];
%*********************************************************************
yy(:,1) = yy(:,1)/max(yy(:,1));yy(:,2) = yy(:,2)/max(yy(:,2));
Y1 = [];UU = [];
%-------------------------------------------------------------------
for i=1:N
    [Hy1, HH1] = ActiveDesign3(PtrTrain,PP,hyp1,hyp01);
    [Hy2, HH2] = ActiveDesign3(PtrTrain,PP,hyp2,hyp02);
    for ii=1:length(PP(:,1))
        UU(ii,:) = [sqrt(Hy1(ii)^2+Hy2(ii)^2)...
            sqrt(yy(ii,1)^2+yy(ii,2)^2)];
    end
    maxHy = [HH1 HH2];
    %-------------------------------------------------------------------
    [Front1,Fronts] = FrontsSortOnline(UU);
    PU = [];U = [];
    for mm=1:length(Front1)
        PU(mm,:) = [PP(Front1(mm),:) Front1(mm)];
        U(mm,:) = UU(Front1(mm),:);
    end
    PUM = PU(:,1:end-1);
    test = zeros(length(PU(:,1)),1);
    for iv=1:length(PU(:,1))
        test(iv) = U(iv,1)*(sqrt(2)-(U(iv,2)));
    end
    [hh,indtest] = max(test);
    Best = UtilityMaxAckley(PUM,U,PP,PtrTrain,range,ytrTrain,...
        0.001,MAX,Target,hyp1,hyp01,hyp2,hyp02,maxHy);
    s1 = ackleyMatrix(Best);
    s2 = boothMatrix(Best);
    Y1 = [Y1;s1 s2];
    PtrTrain = [PtrTrain;Best];
    total = total + 1;
    ytrTrain = [ytrTrain;Y1];
    count = count + 1;
    if (Best(1)==PU(indtest,1)) && (Best(2)==PU(indtest,2))
        disp('same one')
        PP(PU(indtest,end),:) = [];
    end
end
r11 = max(ytrTrain(:,1))-Target(1);
r22 = max(ytrTrain(:,2))-Target(2);
if r11>r1
```

```
    r1 = r11;
end
if r22>r2
    r2 = r22;
end
v = trial1(ytrTrain,Target,r1,r2);
RunV = [RunV v];
%-------------------------------------------------------------
figure(10)
plot(Best(1),Best(2),'k^','MarkerFaceColor','k','markersize',6)
hold on
figure(11)
plot(Best(1),Best(2),'k^','MarkerFaceColor','k','markersize',6)
hold on
%*****************************************************************
C1 = Y1;
figure(2)
plot(C1(:,1),C1(:,2),'k^','MarkerFaceColor','k','markersize',6);
hold on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=2:NI
    it = h;
    if ((RunV(it)-RunV(it-1))/RunV(it-1))<0.01
        stop = stop +1;
        if stop==4
            PP = PLOTgauss2D(PtrTrain(end-count+1:end,:),1200);
            PtrTrain = PtrTrain(end-count+1:end,:);
            ytrTrain = ytrTrain(end-count+1:end,:);
            stop = 0;count = length(PtrTrain(:,1));
        end
    end
    id = [1,1];
    sn1 = 0.01;sn2 = 0.01;
    sf1 = 1;
    sf2 = 1;
    cov  = {'covMaterniso',3};
    ell1 = 2;
    ell2 = 2;
    hyp01.cov  = log([ell1 sf1]);
    hyp02.cov  = log([ell2 sf2]);
    mean = {@meanZero};
    lik_list = {'likGauss','likLaplace','likSech2','likT'};
    inf_list = {'infExactNew','infLaplace','infEP','infVB'};
    for i=1:size(id,1)
        lik = lik_list{id(i,1)};
        hyp01.lik  = log(sn1);
```

```
    hyp02.lik  = log(sn2);
    inf = inf_list{id(i,2)};
end
%"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
[Fmax1,I1,fmin1,best1] = findMaxLik(PtrTrain,ytrTrain(:,1),...
    15,1,60);
[Fmax2,I2,fmin2,best2] = findMaxLik(PtrTrain,ytrTrain(:,2),...
    15,1,60);
hyp1.lik  = fmin1(I1,2);
hyp2.lik  = fmin2(I2,2);
hyp1.cov  = fmin1(I1,3:end);
hyp2.cov  = fmin2(I2,3:end);
%************************************
counter1 = crossVmodel(PtrTrain,ytrTrain(:,1),hyp1);
if counter1~=0
    disp('model 1 is no good!')
end
counter2 = crossVmodel(PtrTrain,ytrTrain(:,2),hyp2);
if counter2~=0
    disp('model 2 is no good!')
end
[ymu1{1}, ys21{1}] = gpCorrected(hyp1, inf, mean, cov, lik,...
    PtrTrain, ytrTrain(:,1),PP);  % predict
if isempty(ymu1{1})==1 || isempty(ys21{1})==1
    [ymu1{1}, ys21{1}] = gpCorrected(hyp01, inf, mean, cov, lik,...
        PtrTrain, ytrTrain(:,1),PP);  % predict
end
[ymu2{1}, ys22{1}] = gpCorrected(hyp2, inf, mean, cov, lik,...
    PtrTrain, ytrTrain(:,2),PP);  % predict
if isempty(ymu2{1})==1 || isempty(ys22{1})==1
    [ymu2{1}, ys22{1}] = gpCorrected(hyp02, inf, mean, cov, lik,...
        PtrTrain, ytrTrain(:,2),PP);  % predict
end
ym1 = ymu1{1};ym2 = ymu2{1};
s = [ym1 ym2];
yy = mgaFunctions(s,Target);
MAX = [max(yy(:,1)) max(yy(:,2))];
%*****************************************************************
yy(:,1) = yy(:,1)/max(yy(:,1));
yy(:,2) = yy(:,2)/max(yy(:,2));
%*****************************************************************
Y1 = [];
UU = [];
%-----------------------------------------------------------------
for i=1:N
    [Hy1, HH1] = ActiveDesign3(PtrTrain,PP,hyp1,hyp01);
```

```
    [Hy2, HH2] = ActiveDesign3(PtrTrain,PP,hyp2,hyp02);
    for ii=1:length(PP(:,1))
        UU(ii,:) = [sqrt(Hy1(ii)^2+Hy2(ii)^2)...
            sqrt(yy(ii,1)^2+yy(ii,2)^2)];
    end
    maxHy = [HH1 HH2];
    %-------------------------------------------------------------------------
    [Front1,Fronts] = FrontsSortOnline(UU);
    PU = [];U = [];
    for mm=1:length(Front1)
        PU(mm,:) = [PP(Front1(mm),:) Front1(mm)];
        U(mm,:) = UU(Front1(mm),:);
    end
    PUM = PU(:,1:end-1);
    test = zeros(length(PU(:,1)),1);
    for iv=1:length(PU(:,1))
        test(iv) = U(iv,1)*(sqrt(2)-(U(iv,2)));
    end
    [hh,indtest] = max(test);
    Best = UtilityMaxAckley(PUM,U,PP,PtrTrain,range,ytrTrain,...
        0.001,MAX,Target,hyp1,hyp01,hyp2,hyp02,maxHy);
    s1 = ackleyMatrix(Best);
    s2 = boothMatrix(Best);
    Y1 = [Y1;s1 s2];
    PtrTrain = [PtrTrain;Best];
    total = total + 1;
    ytrTrain = [ytrTrain;Y1];
    if (Best(1)==PU(indtest,1)) && (Best(2)==PU(indtest,2))
        disp('same one')
        PP(PU(indtest,end),:) = [];
    end
end
%-------------------------------------------------------------------
figure(10)
plot(Best(1),Best(2),'k^','MarkerFaceColor','k','markersize',6)
hold on
figure(11)
plot(Best(1),Best(2),'k^','MarkerFaceColor','k','markersize',6)
hold on
%-------------------------------------------------------------------
C1 = Y1;
figure(2)
plot(C1(:,1),C1(:,2),'k^','MarkerFaceColor','k','markersize',6);
hold on
%-----------------------------------------------
r11 = max(ytrTrain(:,1))-Target(1);
```

```
    r22 = max(ytrTrain(:,2))-Target(2);
    if r11>r1
        r1 = r11;
    end
    if r22>r2
        r2 = r22;
    end
    v = trial1(ytrTrain,Target,r1,r2);
    RunV = [RunV v];
end
h30b = figure(10);
print(h30b,'-deps2','Figure30bM5.eps')
h30a = figure(11);
print(h30a,'-deps2','Figure30aM5.eps')
h30pp = figure(11);
print(h30pp,'-deps2','Figure30pp.eps')
v = trial1(ytrTrain,Target,r1,r2);


function y = ackleyMatrix(P)

% Function for matrix input.
% Input: P - matrix where each row is a candidate solution
% Output: y - a vector of the Ackley function values

N = length(P(:,1));
y = zeros(N,1);
for i=1:N
    y(i) = ackley(P(i,:));
end


function y = boothMatrix(P)

% Function for matrix input
% Input: P - matrix where each row is a candidate solution
% Output: y - a vector of the Booth function values

N = length(P(:,1));
y = zeros(N,1);
for i=1:N
    y(i) = booth(P(i,:));
end
```

```
function y = ackley(x)

% Ackley function.
% Input: x — candidate solution
% Output: y — the Ackley function value

n = length(x);
a = 20; b = 0.2; c = 2*pi;
s1 = 0; s2 = 0;
for i=1:n;
   s1 = s1+x(1,i)^2;
   s2 = s2+cos(c*x(1,i));
end
y = -a*exp(-b*sqrt(1/n*s1))-exp(1/n*s2)+a+exp(1);
y = y + randn(1,1)*0.01;


function y = booth(x)

% The Booth function
% Input: x — candidate solution
% Output: y — the Booth function value
% The number of variables n = 2.
%
y  = (x(1)+2*x(2)-7)^2+(2*x(1)+x(2)-5)^2;
if y==0
    y = 0.001;
    y = log(y);
    y = y + randn(1,1)*0.001;
else
    y = log(y);
    y = y + randn(1,1)*0.001;
    if exp(y)<0.001
        y = log(0.001);
    end
end


function y = ackleyMatrixContour(X,Y)

% Function to produce a contour plot of the Ackley function
% Input: X,Y — vectors of input variables values (for the mesh grid)
% Output: y — the corresponding (for the mesh grid) function values

N = length(X);
```

```
y = zeros(N,N);
for i=1:length(X(1,:))
    for j=1:length(X(:,1))
        y(i,j) = ackley([X(i,j) Y(i,j)]);
    end
end


function y = boothMatrixContour(X,Y)

% Function to produce a contour plot of the Booth function
% Input: X,Y - vectors of input variables values (for the mesh grid)
% Output: y - the corresponding (for the mesh grid) function values

N = length(X);
y = zeros(N,N);
for i=1:length(X(1,:))
    for j=1:length(X(:,1))
        y(i,j) = booth([X(i,j) Y(i,j)]);
    end
end


function [Fmax,I,fmin] = findMaxLik(x,y,N,range)

% Function to perform a number of random starts for the computation of
% maximum likelihood
% Input: x - matrix of candidate solutions
%        y - vector of corresponding observations
%        N - number of random starts
%        range - the range of the input variable values
% Output: Fmax - the best maximum likelihood value from all of the runs
%         fmin - N x 4 matrix where each row contains hyperparameter values
%         corresponding with each one of N random starts
%         I - row number of the best run

initial = ones(N,1);
for n=1:N
    r = 1 + sqrt(range)*rand;
    initial(n,:) = initial(n,:)*r;
end
fmin = zeros(N,4);
for h=1:N
    id = [1,1];
    sn = 0.01;
```

```
    cov  = {'covMaterniso',3};
    sf = 1;
    hyp0.cov  = log([initial(h,:) sf]);
    mean = {@meanZero};
    lik_list = {'likGauss','likLaplace','likSech2','likT'};
    inf_list = {'infExactNew','infLaplace','infEP','infVB'};
    Ncg = 50;
    nlZ1(1) = -Inf;
    for i=1:size(id,1)
        lik = lik_list{id(i,1)};
        if strcmp(lik,'likT')
            nu = 4;
            hyp0.lik  = log([nu-1;sqrt((nu-2)/nu)*sn]);
        else
            hyp0.lik  = log(sn);
        end
        inf = inf_list{id(i,2)};
        if Ncg==0
            hyp1 = hyp0;
        else
            [hyp1, fX, i1] = minimize(hyp0,'gpCorrected', -Ncg, inf,...
                mean, cov, lik,x,y);
        end
        [nlZ1(1) dnlZ1(1)] = gpCorrected(hyp1, inf, mean, cov, lik,x,y);
        nlz1 = nlZ1(1);
        fmin(h,:) = [nlz1 hyp1.lik hyp1.cov];

    end
end
%=========================================================================
[Fmax,I] = max(fmin(:,1));


function [counter] = crossVmodel(C,s,hyp00)

% Function to perform a GP model validation
% Input: C - set of solutions of the training set
%        s - set of corresponding observations
%        hyp00 - hyperparameters of the GP model
% Output: counter - number of training samples that failed validation

L = length(C(:,1));
Sr = zeros(L,1);
counter = 0;

trainIt1 = zeros(L-1,2);Strain = zeros(L-1,1);
```

```
c1 = cvpartition(length(C(:,1)),'leaveout');
for p=1:L
    idxTrain1 = training(c1,p);con = 1;s1 = 0;
    for jj=1:L
        if idxTrain1(jj)==1
            trainIt1(con,:) = C(jj,:);
            Strain(con) = s(jj);
            con = con + 1;
        else
            testIt1 = C(jj,:);
            s1 = s(jj);
        end
    end
    [ymu1,ys21] = GPtest(trainIt1,Strain,testIt1,hyp00);
    Sr(p) = (s1-ymu1)/sqrt(ys21);
    if (Sr(p)<-3) || (Sr(p)>3)
        counter = counter + 1;
    end
end


function [ymu1,ys21] = GPtest(P,s,Pnew,hyp00)

% Function to run the GP model being validated for various folds during
% crossvalidation
% Input: P - matrix of candidate solutions
%        s - corresponding observations
%        Pnew - a candidate solution for which a prediction is to be made
% Output: ymu1 - predicted mean value for Pnew
%         ys21 - predicted variance for Pnew

id = [1,1];
sn1 = 0.01;sf1 = 1;
cov  = {'covMaterniso',3};
ell1 = 2;
hyp01.cov  = log([ell1 sf1]);
mean = {@meanZero};
lik_list = {'likGauss','likLaplace','likSech2','likT'};
inf_list = {'infExactNew','infLaplace','infEP','infVB'};
for i=1:size(id,1)
    lik = lik_list{id(i,1)};
    hyp01.lik  = log(sn1);
    inf = inf_list{id(i,2)};
end
%"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
[ymu1{1}, ys21{1}] = gpCorrected(hyp00, inf, mean, cov, lik, P,s,Pnew);
```

```
if isempty(ymu1{1})==1 || isempty(ys21{1})==1
    [ymu1{1}, ys21{1}] = gpCorrected(hyp01, inf, mean, cov, lik,...
        P,s,Pnew);
end
ymu1 = ymu1{1};ys21 = ys21{1};


function [Hy,HH] = ActiveDesign3(Ptr,G,hyp1,hyp0)

% Function to compute the marginal increase in mutual information
% Input: Ptr — solutions from the training set
%        G — candidate solutions from the remainder of the grid
%        hyp1,hyp0 — hyperparameters of the current GP model
% Output: Hy — a vector of marginal increases for solutions in G
%         HH — maximum value in Hy

LL = length(Ptr(:,1));
id = [1,1];
cov  = {'covMaterniso',3};
mean = {@meanZero};
lik_list = {'likGauss','likLaplace','likSech2','likT'};
inf_list = {'infExactNew','infLaplace','infEP','infVB'};
for i=1:size(id,1)
    lik = lik_list{id(i,1)};
    inf = inf_list{id(i,2)};
end
[ymu1{1}, ys21{1}] = gpCorrected(hyp1, inf, mean, cov, lik, Ptr,...
    ones(length(Ptr(:,1)),1),G);  % predict
if isempty(ymu1{1})==1 || isempty(ys21{1})==1
    [ymu1{1}, ys21{1}] = gpCorrected(hyp0, inf, mean, cov, lik, Ptr,...
        ones(length(Ptr(:,1)),1),G);  % predict
end
ys1 = ys21{1};
K = covMaterniso(3,hyp1.cov, G);
Hy = zeros(1,length(G(:,1)));
for i=1:length(G(:,1))
    v = zeros(length(G(:,1))-1,length(G(1,:))+1);n = 1;
    %-------------------------------------------------------------
    for j=1:length(G(:,1))
        if i~=j
            v(n,:) = [G(j,:) abs(K(i,j))];n = n + 1;
        end
    end
    [ee1,ind1] = sort(v(:,end),'descend');
    if 2*LL>length(v(:,1))
        v1 = zeros(length(v(:,1)),length(G(1,:)));
```

```
    else
        v1 = zeros(2*LL,length(G(1,:)));
    end
    for m=1:2*LL
        v1(m,:) = v(ind1(m),1:end-1);
    end
    [ymu1{1}, ys31{1}] = gpCorrected(hyp1, inf, mean, cov, lik, v1,...
        ones(length(v1(:,1)),1),G(i,:));  % predict
    if isempty(ymu1{1})==1 || isempty(ys21{1})==1
        [ymu1{1}, ys31{1}] = gpCorrected(hyp0, inf, mean, cov, lik, v1,...
            ones(length(v1(:,1)),1),G(i,:));  % predict
    end
    if isempty(ys31{1})==1
        Hy(i) = -Inf;
    else
        ys11 = ys31{1};
        Hy(i) = log(ys1(i)/ys11);
    end
end
HH = max(Hy);
Hy = Hy/(max(Hy));


function [Front1,Fronts] = FrontsSortOnline(X)

% Function to sort solutions (in fitness space) into fronts.
% Input: X - input matrix of the solutions' fitness, where columns are
% respective objective's observation.
% Output: Fronts - output matrix where each row is a front
% (in ascending order).
%         Front1 - the leading front (Paretto solutions)

Fronts = zeros(length(X(:,1)),length(X(:,1)));
p = 1;Front1 = [];
for i=1:length(X(:,1))
    n(i) = 0;q = 1;
    for j=1:length(X(:,1))
        if i ~= j
            c = 0;
            if (X(i,1) >= X(j,1)) || (X(i,2) <= X(j,2))
                c = c + 1;
            end
            if c == 0
                n(i) = n(i) + 1;
            end
        end
```

```
    end
    if n(i) == 0
        Fronts(1,p) = i;p = p + 1;
        Front1 = [Front1 i];
    end
end
n = zeros(1,length(X(:,1)));cc = 1;
while (sum(n) ~= -1*length(n))
    p = 1;
    for h=1:length(n)
        for h1=1:length(n)
            if Fronts(cc,h) ~= 0 && Fronts(cc,h) == h1
                n(h1) = -1;
            end
        end
    end
    for i=1:length(X(:,1))
        for j=1:length(X(:,1))
            if i ~= j && n(j) ~= -1 && n(i) ~= -1
                c = 0;
                if (X(i,1) >= X(j,1)) || (X(i,2) <= X(j,2))
                    c = c + 1;
                end
                if c == 0
                    n(i) = n(i) + 1;
                end
            end
        end
        if n(i) == 0
            Fronts(cc + 1,p) = i;
            p = p + 1;
        end
    end
    cc = cc + 1;
    for h2=1:length(n)
        if n(h2) ~= -1
            n(h2) = 0;
        end
    end
end


function Best = UtilityMaxAckley(PU,U,PP,PtrTrain,range,ytrTrain,...
    epsilon,maxYY,Target,hyp10,hyp01,hyp20,hyp02,maxHy)

% Function to search for the next best candidate solution to evaluate
```

```
% (using Genetic Algorithm)
% Input: U - matrix of fitness, first row-Hy1, second-ym1.
%        PU - matrix of formulations for initial population.
%        PP - matrix of the candidate solutions from the grid
%        PtrTrain - the set of solutions collected so far
%        ytrTrain - the set of observations collected so far
%        range - the range of the input variables' values
%        epsilon - parameter for the computation of pertrubations during
%        mutation step of the GA
%        maxYY - maximum predicted means of the solutions on the grid
%        (for all of the approximated processes)
%        maxHy - maximum predicted marginal increases in mutual information
%        of the solutions on the grid (for all of the approximated processes)
%        Target - vector of target values
%        hyp10,hyp01,hyp20,hyp02 - hyperparameters of the GP models
% Output: Best - the next candidate solution to evaluate

test = zeros(length(U(:,1)),1);
for i=1:length(U(:,1))
    test(i) = U(i,1)*(sqrt(2)-(U(i,2)));
end
[hh,indtest] = max(test);
bestVol = hh;
Best = PU(indtest,:);
for lp=1:length(PP(:,1))
    if PP(lp,1)==Best(1) && PP(lp,2)==Best(2)
        num = lp;
    end
end
PP(num,:) = [];
I = indtest;
id = [1,1];
cov  = {'covMaterniso',3};
mean = {@meanZero};
lik_list = {'likGauss','likLaplace','likSech2','likT'};
inf_list = {'infExactNew','infLaplace','infEP','infVB'};
for i=1:size(id,1)
    lik = lik_list{id(i,1)};
    inf = inf_list{id(i,2)};
end
%-------------------------------------------------------------------------
O = InitialApproachOnline(I,PU,range,U,epsilon,0);
[ymu1{1}, ys21{1}] = gpCorrected(hyp10, inf, mean, cov, lik, PtrTrain,...
    ytrTrain(:,1),O);  % predict
if isempty(ymu1{1})==1 || isempty(ys21{1})==1
    [ymu1{1}, ys21{1}] = gpCorrected(hyp01, inf, mean, cov, lik,...
```

```
        PtrTrain, ytrTrain(:,1),O);  % predict
end
[ymu2{1}, ys22{1}] = gpCorrected(hyp20, inf, mean, cov, lik, PtrTrain,...
    ytrTrain(:,2),O);  % predict
if isempty(ymu2{1})==1 || isempty(ys22{1})==1
    [ymu2{1}, ys22{1}] = gpCorrected(hyp02, inf, mean, cov, lik,...
        PtrTrain, ytrTrain(:,2),O);  % predict
end
ym1 = ymu1{1};ym2 = ymu2{1};
Hy1 = ActiveDesignOnlineAckley(PtrTrain,PP,O,hyp10,hyp01,maxHy(1));
Hy2 = ActiveDesignOnlineAckley(PtrTrain,PP,O,hyp20,hyp02,maxHy(2));
yy = mgaFunctions([ym1 ym2],Target);
yy(:,1) = yy(:,1)/maxYY(1);yy(:,2) = yy(:,2)/maxYY(2);
U1 = [];
for i=1:length(O(:,1))
    U1(i,:) = [sqrt(Hy1(i)^2+Hy2(i)^2) sqrt(yy(i,1)^2+yy(i,2)^2)];
end
test = zeros(length(U1(:,1)),1);
for i=1:length(U1(:,1))
    test(i) = U1(i,1)*(sqrt(2)-(U1(i,2)));
end
[hh,indtest1] = max(test);
if hh>bestVol
    bestVol = hh;
    Best = O(indtest1,:);
    I = indtest1;
end
%-------------------------------------------------------------------------
Past = PU;New = O; PastS = U; NewS = U1;
for h=1:100
    [offsprings] = ElitistApproachOnline(I,Past,New,PastS,NewS,range,...
        epsilon,0);
    Po = offsprings;
    [ymu1{1}, ys21{1}] = gpCorrected(hyp10, inf, mean, cov, lik,...
        PtrTrain, ytrTrain(:,1),Po);  % predict
    if isempty(ymu1{1})==1 || isempty(ys21{1})==1
        [ymu1{1}, ys21{1}] = gpCorrected(hyp01, inf, mean, cov, lik,...
            PtrTrain, ytrTrain(:,1),Po);  % predict
    end
    [ymu2{1}, ys22{1}] = gpCorrected(hyp20, inf, mean, cov, lik,...
        PtrTrain, ytrTrain(:,2),Po);  % predict
    if isempty(ymu2{1})==1 || isempty(ys22{1})==1
        [ymu2{1}, ys22{1}] = gpCorrected(hyp02, inf, mean, cov,...
            lik,PtrTrain, ytrTrain(:,2),Po);  % predict
    end
    ym1 = ymu1{1};ym2 = ymu2{1};
```

```
    Hy1 = ActiveDesignOnlineAckley(PtrTrain,PP,Po,hyp10,hyp01,maxHy(1));
    Hy2 = ActiveDesignOnlineAckley(PtrTrain,PP,Po,hyp20,hyp02,maxHy(2));
    yy = mgaFunctions([ym1 ym2],Target);
    yy(:,1) = yy(:,1)/maxYY(1);yy(:,2) = yy(:,2)/maxYY(2);
    U1 = [];
    for i=1:length(Po(:,1))
        U1(i,:) = [sqrt(Hy1(i)^2+Hy2(i)^2) sqrt(yy(i,1)^2+yy(i,2)^2)];
    end
    test = zeros(length(U1(:,1)),1);
    for i=1:length(Po(:,1))
        test(i) = U1(i,1)*(sqrt(2)-(U1(i,2)));
    end
    [hh,indtest2] = max(test);
    if hh>bestVol
        bestVol = hh;
        Best = Po(indtest2,:);
        I = indtest2;
    end
    Past = New; New = Po;PastS = NewS;NewS = U1;
end


function [offsprings] = InitialApproachOnline(I,P,range,p1,epsilon,Crate)

% Function to run the GA for the first population.
% Input: I - the best candidate solutions so far.
%        P - matrix containing the first population of candidate solutions
%        range - the range of the input variables' values
%        p1 - vector of observations for the first population of candidate
%        solutions
%        epsilon - parameter for the computation of pertrubations during
%        mutation step
%        Crate - probability of crossover (always 0).
% Output: offsprings - candidate solutions of the newly produced population

[Front1,Fronts] = FrontsSortOnline(p1);
A = CrowdingDistanceAssignment(p1,Fronts);
f = Ranking(Fronts,A,length(P(:,1)));
Select = Selection11(I,f);
[offsprings] = CrossoverAndMutationRatioOnlinePOLinput(P,Select,...
    length(P(1,:)),range,epsilon,Crate);


function [offsprings] = ElitistApproachOnline(I,P,O,p,p1,range,epsilon,Crate)
```

```
% Function to construct new population of the GA using elitest approach.
% Input: I — the best candidate solutions so far.
%        P — input matrix of candidate solutions from the preceeding
%        population.
%        O — input matrix of candidate solutions of the current
%        (offspring) population.
%        p — input matrix of observations of solutions from
%        the preceeding population.
%        p1 — input matrix of observations of solutions from
%        the current population.
%        range — the range of the input variables' values
%        epsilon — parameter for the computation of pertrubations during
%        mutation step
%        Crate — probability of crossover (always 0).
% Output: offsprings — candidate solutions of the newly produced population

R = [p;p1];
R1 = [P;O];
[Front1,Fronts] = FrontsSortOnline(R);
A = CrowdingDistanceAssignment(R,Fronts);
f = Ranking(Fronts,A,length(P(:,1)));
ranked = f(1:length(P(:,1)),1);
Select = Selection11(I,ranked);
[offsprings] = CrossoverAndMutationRatioOnlinePOLinput(R1,Select,...
    length(P(1,:)),range,epsilon,Crate);


function A = CrowdingDistanceAssignment(X,Fronts)

% This function calculates the average distance of every candidate solution
%(in the fitness space) from the solutions in the same front.
% Input: X — input matrix of candidate solutions (rows — solutions,
%            columns — fitness values).
%        Fronts — input matrix where each row is a front
%        (in ascending order).
% Output: A — matrix containing average distance for each solution
%         (from the points in the same front).

A = Fronts;
for i=1:length(Fronts(:,1))
    for j=1:length(Fronts(1,:))
        TotalDistance = 0;
        if Fronts(i,j) ~= 0
            counter = 0;
            for k=1:length(Fronts(1,:))
                if Fronts(i,k) ~= 0
```

```
                    D = 0;counter = counter + 1;
                    for l=1:length(X(1,:))
                        d = (X(Fronts(i,j),l) - X(Fronts(i,k),l))^2;
                        D = D + d;
                    end
                    s = sqrt(D);TotalDistance = TotalDistance + s;
                end
            end
            if (counter - 1)==0
                A(i,j)=0.0000000001;
            else
                A(i,j) = TotalDistance/(counter - 1);
            end
        end

    end
end


function f = Ranking(Fronts,A,Ninputs)

% Function that performs ranking and scaling on the population (in fitness
% space).
% Input: Fronts - input matrix where each row is a front
%        (in ascending order).
%        A - input matrix containing average distance for each solution
%        (from the points in the same front).
%        Ninputs - number of solutions in the population
% Output: f - output matrix of ranked solutions.

f = zeros(1,Ninputs);c = 1;
for i=1:length(Fronts(:,1))
    N = A(i,:);
    bb = length(Fronts(1,:)) + 1;
    for ii=1:length(N)
        if N(ii) == 0
            bb = ii;break;
        end
    end
    N = N(1:bb-1);
    if isempty(N) == 0
        [b,e] = sort(N,'descend');
        for k=1:length(N)
            f(c) = Fronts(i,e(k)); c = c + 1;
        end
    end
```

```
end
f = f';


function P = Selection11(I,f)

% Function that selects P parents (candidate solutions) for reproduction.
% I – the best candidate solutions so far.
% f – an array of ranked candidate solutions' indices arranged in ascending
% order (the first one is the fittest).

P = []; M = [];                 % initialize the mating  and tournament pool.
m = randi(length(f),1,2);       % create the first tournament.
M = [M;m];
if m(1,1) < m(1,2)
    P(1) = f(m(1,1));
else
    P(1) = f(m(1,2));                       % save the first parent.
end

for i=2:length(f)
    pair = randi(length(f),1,2);
    while pair(1,1) == pair(1,2)
        pair(1,:) = randi(length(f),1,2);
    end
    counter = 0;
    while counter == 0
        for j=1:length(M(:,1))
            if sum(ismember(pair,M(j,:))) == 2
                counter = 1;break;
            end
        end
        if counter == 1
            pair = randi(length(f),1,2);
            while pair(1,1) == pair(1,2)
                pair(1,:) = randi(length(f),1,2);
                counter = 0;
            end
        else
            counter = 1;
        end
    end
    M = [M;pair];
    if pair(1,1) < pair(1,2)
        P(i) = f(pair(1,1));
    else
```

```
        P(i) = f(pair(1,2));                    % save the parent.
    end
end


function [offsprings] = CrossoverAndMutationRatioOnlinePOLinput...
    (X,r,n,range,epsilon,Crate)

% Function to perform mutations (perturbations) on the candidate solutions
% Input: X — matrix containing the current population of candidate solutions
%        range — the range of the input variables' values
%        r — indices of solutions selected for reproduction
%        n — number of input variables in a solution
%        epsilon — parameter for the computation of pertrubations during
%        mutation step
%        Crate — probability of crossover (always 0).
% Output: offsprings — candidate solutions of the newly produced population

offsprings = zeros(length(r),n);w1 = 1;
for tt=1:length(r)
    epsilon1 = epsilon*rand(1);
    epsilon2 = epsilon*rand(1);
    %-----------------------------------------
    c1 = X(r(tt),:);
    if rand(1)>0.5
        c1(1) = c1(1) + epsilon1;
        if c1(1)>1
            c1(1) = X(r(tt),1);
        end
    else
        c1(1) = c1(1) - epsilon1;
        if c1(1)<0
            c1(1) = X(r(tt),1);
        end
    end
    %*************************************
    if rand(1)>0.5
        c1(2) = c1(2) + epsilon2;
        if c1(2)>1
            c1(2) = X(r(tt),2);
        end
    else
        c1(2) = c1(2) - epsilon2;
        if c1(2)<0
            c1(2) = X(r(tt),2);
        end
```

```
    end
    %-----------------------------------------
    offsprings(w1,:) = c1;
    w1 = w1 + 1;
end


function [Hy] = ActiveDesignOnlineAckley(Ptr,G,O,hyp1,hyp0,maxHy)

% Function to compute the marginal increase in mutual information for a
% current population of candidate solutions selected by GA
% Input: Ptr — solutions from the training set
%        G — candidate solutions from the remainder of the grid
%        O — candidate solutions selected By the GA
%        hyp1,hyp0 — hyperparameters of the current GP model
%        maxHy — maximum predicted marginal increases in mutual information
%        of the solutions on G (for all of the approximated processes)
% Output: Hy — a vector of marginal increases for solutions in O


LL = length(Ptr(:,1));
P = G;
id = [1,1];
cov  = {'covMaterniso',3};
mean = {@meanZero};
lik_list = {'likGauss','likLaplace','likSech2','likT'};
inf_list = {'infExactNew','infLaplace','infEP','infVB'};
for i=1:size(id,1)
    lik = lik_list{id(i,1)};
    inf = inf_list{id(i,2)};
end
[ymu1{1}, ys21{1}] = gpCorrected(hyp1, inf, mean, cov, lik, Ptr,...
    ones(length(Ptr(:,1)),1),O);  % predict
if isempty(ymu1{1})==1 || isempty(ys21{1})==1
    [ymu1{1}, ys21{1}] = gpCorrected(hyp0, inf, mean, cov, lik, Ptr,...
        ones(length(Ptr(:,1)),1),O);  % predict
end
ys1 = ys21{1};
Hy = zeros(1,length(O(:,1)));
for i=1:length(O(:,1))
    K = covMaterniso(3,hyp1.cov,[O(i,:);P]);
    v = zeros(length(P(:,1)),length(P(1,:))+1);n = 1;
    %-------------------------------------------------------------
    for j=1:length(P(:,1))
        v(n,:) = [P(j,:) abs(K(1,j+1))];n = n + 1;
    end
```

```
    [ee1,ind1] = sort(v(:,end),'descend');
    v1 = zeros(2*LL,length(P(1,:)));
    for m=1:2*LL
        v1(m,:) = v(ind1(m),1:end-1);
    end
    [ymu1{1}, ys31{1}] = gpCorrected(hyp1, inf, mean, cov, lik, v1,...
        ones(length(v1(:,1)),1),O(i,:));  % predict
    if isempty(ymu1{1})==1 || isempty(ys21{1})==1
        [ymu1{1}, ys31{1}] = gpCorrected(hyp0, inf, mean, cov, lik, v1,...
            ones(length(v1(:,1)),1),O(i,:));  % predict
    end
    if isempty(ys31{1})==1
        Hy(i) = -Inf;
    else
        ys11 = ys31{1};
        Hy(i) = log(ys1(i)/ys11);
    end
end
Hy = Hy/maxHy;


function v = trial1(ytrTrain,Target,rr1,rr2)

% Function to compute the value of the hypervolume indicator for the current
%         Paretto front
% Input: ytrTrain - the set of observations collected so far
%        Target - vector of target values
%        rr1,rr2 - current ranges of the observed values of the
%        approximated processes
% Output: v - the value of a hypervolume indicator for the current
%         Paretto front

ytrTrain1 = mgaFunctions(ytrTrain,Target);
y01 = max(ytrTrain1(:,1));y02 = max(ytrTrain1(:,2));
y1 = [ytrTrain1(:,1)/y01 ytrTrain1(:,2)/y02];
[currentFront1Matrix] = currentParettoFront(y1,...
    length(y1(1,:)));
v=hypervolume(currentFront1Matrix,[1 1],1000000);


function [FrontsMatrix] = currentParettoFront(X,Objs)

% Function to compute the Paretto front only.
% X - input matrix of the solutions' fitness, where columns are
% respective objective's solution.
```

```
% Objs — number of objectives
% FrontsMatrix — Paretto front.

Fronts = zeros(1,length(X(:,1)));
FrontsMatrix = [];
p = 1;
for i=1:length(X(:,1))
    n(i) = 0;
    for j=1:length(X(:,1))
        if i ~= j
            c = 0;
            for k=1:Objs
                if X(i,k) < X(j,k)
                    c = c + 1;
                end
            end
            if c == 0
                n(i) = n(i) + 1;
            end
        end
    end
    if n(i) == 0
        Fronts(p) = i;p = p + 1;
        FrontsMatrix = [FrontsMatrix;X(i,:)];
    end
end


function v=hypervolume(P,r,N)

% Function to compute the value of hypervolume indicator as
% a measure of Paretto front estimate (as a persentage).
% Input: P — Pareto front
%        r — reference point
%        N — number of uniformly distributed random points within
%   the bounded hyper—cuboid
% Output: v — estimate of the hypervoulme


% Check input and output
error(nargchk(2,3,nargin));
error(nargoutchk(0,1,nargout));

P=P*diag(1./r);
[n,d]=size(P);
if nargin<3
```

```
    N=1000;
end
if ~isscalar(N)
    C=N;
    N=size(C,1);
else
    C=rand(N,d);
end

fDominated=false(N,1);
lB=min(P);
fcheck=all(bsxfun(@gt, C, lB),2);

for k=1:n
    if any(fcheck)
        f=all(bsxfun(@gt, C(fcheck,:), P(k,:)),2);
        fDominated(fcheck)=f;
        fcheck(fcheck)=~f;
    end
end

v=sum(fDominated)/N;



function Y11 = PLOTgauss2D(PtrTrain,N1)

% Function to resample the grid
% Input: PtrTrain – the set of solutions collected so far
%        N1 – number of candidate solutions in the new grid
% Output: Y11 – set candidate solutions for the new grid

obj1 = gmdistribution.fit(PtrTrain,2,'Regularize',0.0001);
a1 = obj1.mu;
b1 = obj1.Sigma;
c1 = obj1.PComponents;

Y1 = [mvnrnd(a1(1,:),b1(:,:,1),N1*c1(1));mvnrnd(a1(2,:),b1(:,:,2),...
    N1*c1(2))];
Y11 = [];
for i=1:length(Y1(:,1))
    if (Y1(i,1)>=-30) && (Y1(i,1)<=30) && (Y1(i,2)>=-30) && (Y1(i,2)<=30)
        Y11 = [Y11;Y1(i,:)];
    end
end
plot(PtrTrain(:,1),PtrTrain(:,2),'.')
hold on
```

```
plot(Y11(:,1),Y11(:,2),'r.')
hold off
```

Appendix B

Matlab code for the extension of the MOAL algorithm

The code presented in Appendix B is the one employed for the simulation discussed in chapter 2 section 2.8.3

```
function [PtrTrain, ytrTrain] = OnlineActiveSearchFeas11(zz,AA,PP,N,NI)

% Main function of the extended MOAL algorithm employed for the simulation
% discussed in chapter 2 section 2.8.3
% Input: zz — the vector of target values
%        AA — the training set
%        PP — the set of candidate solutions
%        N  — the number of points to be designed after each iteration
%             (default value is 1)
%        NI — the number of iterations for the optimisation run
% Output: ytrTrain — the set of observations collected during the
%                    optimisation run
%         PtrTrain — the set of corresponding solutions

P11 = zz;
PtrTrain = [];ytrTrain = [];
P1 = [];P2 = [];TR1 = [];TR2 = [];
for i=1:length(PP(:,1))
    ytr1 = 2*((PP(i,1)^2 + 3^PP(i,2)<10) || (PP(i,1)*3 + PP(i,3)^2<10) ...
        || (PP(i,2)*3 + PP(i,3)^2<10)>rand(1,1))-1;
    if (ytr1==1)
        dp11 = dp(PP(i,:));
```

```
        levy11 = levy(PP(i,:));
        P1 = [P1;PP(i,:) 1 dp11 levy11];TR1 = [TR1;PP(i,:)];
    else
        dp11 = dp(PP(i,:));
        levy11 = levy(PP(i,:));
        P2 = [P2;PP(i,:) -1 dp11 levy11];TR2 = [TR2;PP(i,:)];
    end
end
P12 = [P1;P2];
figure(8);
plot3(TR1(:,1),TR1(:,2),TR1(:,3),'c*')
hold on
plot3(TR2(:,1),TR2(:,2),TR2(:,3),'y*')
hold on
plot3(P11(:,1),P11(:,2),P11(:,3),'m.','markersize',20)
A1 = [];A2 = [];
for i=1:length(AA(:,1))
    ytr1 = 2*((PP(i,1)^2 + 3^PP(i,2)<10) || (PP(i,1)*3 + PP(i,3)^2<10) ...
        || (PP(i,2)*3 + PP(i,3)^2<10)>rand(1,1))-1;
    if (ytr1==1)
        dp11 = dp(AA(i,:));
        levy11 = levy(AA(i,:));
        A1 = [A1;AA(i,:) 1 dp11 levy11];
    else
        dp11 = dp(AA(i,:));
        levy11 = levy(AA(i,:));
        A2 = [A2;AA(i,:) -1 dp11 levy11];
    end
end
A12 = [A1;A2];
figure(11);
plot3(A1(:,1),A1(:,2),A1(:,3),'*')
hold on
plot3(A2(:,1),A2(:,2),A2(:,3),'g*')
hold on
PtrTrain = [PtrTrain;A1(:,1:3)];ytrTrain = [ytrTrain;A1(:,5:6)];
figure(2)
plot(ytrTrain(:,1),ytrTrain(:,2),'b.','markersize',10)
title('\it{Measurements Space}','FontSize',16)
xlabel('\it{1st Objective}','FontSize',16)
ylabel('\it{2nd Objective}','FontSize',16)
hold on
figure(2)
plot(P1(:,5),P1(:,6),'c.','markersize',10)
hold on
plot(P2(:,5),P2(:,6),'y.','markersize',10)
```

```
hold on
T1 = dp(P11)';
T2 = levy(P11)';
Target = [T1 T2];
figure(2)
plot(Target(1),Target(2),'m.','markersize',20)
hold on
figure(11);
plot3(P11(:,1),P11(:,2),P11(:,3),'m.','markersize',20)
hold on
[SS,SS1] = GPclassifierFeas(A12,PP);
figure(11);
gg = plot3(SS(:,1),SS(:,2),SS(:,3),'c*');
if isempty(SS1)==0
    hold on
    gg1 = plot3(SS1(:,1),SS1(:,2),SS1(:,3),'y*');
end
id = [1,1];
sn1 = 0.01;sn2 = 0.01;
cov = {@covSEard};sf1 = 1;sf2 = 1;
ell1 = [2 2 2];                                % setup the gp
ell2 = [2 2 2];
hyp01.cov  = log([ell1 sf1]);
hyp02.cov  = log([ell2 sf2]);
mean = {@meanZero};
lik_list = {'likGauss','likLaplace','likSech2','likT'}; %poss likelihoods
inf_list = {'infExactNew','infLaplace','infEP','infVB'};%allowable inf algs
Ncg = 50;                                  % number of conjugate gradient steps
nlZ1(1) = -Inf;
nlZ2(1) = -Inf;
for i=1:size(id,1)
    lik = lik_list{id(i,1)};                          % setup the likelihood
    if strcmp(lik,'likT')
        nu = 4;
        hyp0.lik  = log([nu-1;sqrt((nu-2)/nu)*sn]);
    else
        hyp01.lik  = log(sn1);
        hyp02.lik  = log(sn2);
    end
    inf = inf_list{id(i,2)};
    if Ncg==0
        hyp1 = hyp01;
        hyp2 = hyp02;
    else
        hyp1 = minimize(hyp01,'gpCorrected', -Ncg, inf, mean, cov, lik,...
            PtrTrain,ytrTrain(:,1));
```

```
        hyp2 = minimize(hyp02,'gpCorrected', -Ncg, inf, mean, cov, lik,...
            PtrTrain,ytrTrain(:,2));
    end
end
%-------------------------------------------------------------------------
[ymu1{1}, ys21{1}] = gpCorrected(hyp1, inf, mean, cov, lik, PtrTrain,...
    ytrTrain(:,1),SS);  % predict
[nlZ1(1)] = gpCorrected(hyp1, inf, mean, cov, lik, PtrTrain,ytrTrain(:,1));
[ymu2{1}, ys22{1}] = gpCorrected(hyp2, inf, mean, cov, lik, PtrTrain,...
    ytrTrain(:,2),SS);  % predict
[nlZ2(1)] = gpCorrected(hyp2, inf, mean, cov, lik, PtrTrain,ytrTrain(:,2));
ym1 = ymu1{1};ym2 = ymu2{1};
s = [ym1 ym2];
yy = mgaFunctions(s,Target);
%*************************************************************************
yy(:,1) = yy(:,1)/max(yy(:,1));yy(:,2) = yy(:,2)/max(yy(:,2));
Y1 = [];UU = [];
%-------------------------------------------------------------
for i=1:N
    Hy1 = ActiveDesign3(PtrTrain,SS,hyp1,hyp01);
    Hy2 = ActiveDesign3(PtrTrain,SS,hyp2,hyp02);
    for ii=1:length(SS(:,1))
        UU(ii,:) = [sqrt(Hy1(ii)^2+Hy2(ii)^2) sqrt(yy(ii,1)^2+yy(ii,2)^2)];
    end
    %-------------------------------------------------------------------------
    [Front1,Fronts] = FrontsSortOnline(UU);
    PU = [];U = [];
    for mm=1:length(Front1)
        PU(mm,:) = [SS(Front1(mm),:) Front1(mm)];
        U(mm,:) = UU(Front1(mm),:);
    end
    PUM = PU(:,1:end-1);
    test = zeros(length(PU(:,1)),1);
    for iv=1:length(PU(:,1))
        test(iv) = U(iv,1)*(1-(U(iv,2)));
    end
    [hh,indtest] = max(test);
    Best = UtilityMaxFeas11(PUM,U,SS,PtrTrain,sum(SS(1,:)),...
        ytrTrain,0.01,0.0,Target,hyp1,hyp01,hyp2,hyp02,A12);
    ytr1 = 2*((Best(1)^2 + 3^Best(2)<10) || (Best(1)*3 + Best(3)^2<10)...
        || (Best(2)*3 + Best(3)^2<10)>rand(1,1))-1;
    if (ytr1==1)
        s1 = dp(Best);
        s2 = levy(Best);
        Y1 = [Y1;s1 s2];
        PtrTrain = [PtrTrain;Best];
```

```
        ytrTrain = [ytrTrain;Y1];
        rev = 1;
        PPsame = 0;
        for tr=1:length(PP(:,1))
            if (Best(1)==PP(tr,1)) && (Best(2)==PP(tr,2))...
                    && (Best(3)==PP(tr,3))
                disp('same one')
                PPsame = tr;
                break;
            end
        end
        if PPsame~=0
            PP(PPsame,:) = [];
        end
        A12 = [A12;Best 1 s1 s2];
    else
        dp11 = dp(Best);
        levy11 = levy(Best);
        Y1 = [Y1;dp11 levy11];
        A12 = [A12;Best -1 dp11 levy11];
        total = total + 1;
        rev = 0;
        PPsame = 0;
        for tr=1:length(PP(:,1))
            if (Best(1)==PP(tr,1)) && (Best(2)==PP(tr,2))...
                    && (Best(3)==PP(tr,3))
                disp('same one')
                PPsame = tr;
                break;
            end
        end
        if PPsame~=0
            PP(PPsame,:) = [];
        end
    end
end
C1 = Y1;
if rev==1
    figure(2)
    plot(C1(:,1),C1(:,2),'r.','markersize',10)
    hold on
    figure(11);
    plot3(Best(:,1),Best(:,2),Best(:,3),'r*')
    hold on
else
    disp('black')
```

```
    figure(2)
    plot(C1(:,1),C1(:,2),'k.','markersize',10)
    hold on
    figure(11);
    plot3(Best(:,1),Best(:,2),Best(:,3),'k*')
    hold on
end
for h=2:NI
    [SS,SS1] = GPclassifierFeas(A12,PP);
    figure(11);
    delete(gg);
    gg = plot3(SS(:,1),SS(:,2),SS(:,3),'c*');
    if isempty(SS1)==0
        delete(gg1);
        hold on
        gg1 = plot3(SS1(:,1),SS1(:,2),SS1(:,3),'y*');
    end
    id = [1,1];
    sn1 = 0.01;sn2 = 0.01;
    cov = {@covSEard};sf1 = 1;sf2 = 1;
    ell1 = [2 2 2];
    ell2 = [2 2 2];
    hyp01.cov  = log([ell1 sf1]);
    hyp02.cov  = log([ell2 sf2]);
    mean = {@meanZero};
    lik_list = {'likGauss','likLaplace','likSech2','likT'};
    inf_list = {'infExactNew','infLaplace','infEP','infVB'};
    Ncg = 50;
    nlZ1(1) = -Inf;
    nlZ2(1) = -Inf;
    for i=1:size(id,1)
        lik = lik_list{id(i,1)};
        if strcmp(lik,'likT')
            nu = 4;
            hyp0.lik  = log([nu-1;sqrt((nu-2)/nu)*sn]);
        else
            hyp01.lik  = log(sn1);
            hyp02.lik  = log(sn2);
        end
        inf = inf_list{id(i,2)};
        if Ncg==0
            hyp1 = hyp01;
            hyp2 = hyp02;
        else
            hyp1 = minimize(hyp01,'gpCorrected', -Ncg, inf, mean, cov,...
                lik, PtrTrain,ytrTrain(:,1));
```

```
        hyp2 = minimize(hyp02,'gpCorrected', -Ncg, inf, mean, cov,...
            lik, PtrTrain,ytrTrain(:,2));
    end
end
%"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
[ymu1{1}, ys21{1}] = gpCorrected(hyp1, inf, mean, cov, lik,...
    PtrTrain, ytrTrain(:,1),SS);  % predict
if isempty(ymu1{1})==1 || isempty(ys21{1})==1
    [ymu1{1}, ys21{1}] = gpCorrected(hyp01, inf, mean, cov, lik,...
        PtrTrain, ytrTrain(:,1),SS);  % predict
end
[ymu2{1}, ys22{1}] = gpCorrected(hyp2, inf, mean, cov, lik,...
    PtrTrain, ytrTrain(:,2),SS);  % predict
if isempty(ymu2{1})==1 || isempty(ys22{1})==1
    [ymu2{1}, ys22{1}] = gpCorrected(hyp02, inf, mean, cov, lik,...
        PtrTrain, ytrTrain(:,2),SS);  % predict
end
ym1 = ymu1{1};ym2 = ymu2{1};
s = [ym1 ym2];
yy = mgaFunctions(s,Target);
%*********************************************************************
yy(:,1) = yy(:,1)/max(yy(:,1));yy(:,2) = yy(:,2)/max(yy(:,2));
%*********************************************************************
Y1 = [];
UU = [];
%-------------------------------------------------------------
for i=1:N
    Hy1 = ActiveDesign3(PtrTrain,SS,hyp1,hyp01);
    Hy2 = ActiveDesign3(PtrTrain,SS,hyp2,hyp02);
    for ii=1:length(SS(:,1))
        UU(ii,:) = [sqrt(Hy1(ii)^2+Hy2(ii)^2)...
            sqrt(yy(ii,1)^2+yy(ii,2)^2)];
    end
    %-------------------------------------------------------------------------
    [Front1,Fronts] = FrontsSortOnline(UU);
    PU = [];U = [];
    for mm=1:length(Front1)
        PU(mm,:) = [SS(Front1(mm),:) Front1(mm)];
        U(mm,:) = UU(Front1(mm),:);
    end
    PUM = PU(:,1:end-1);
    test = zeros(length(PU(:,1)),1);
    for iv=1:length(PU(:,1))
        test(iv) = U(iv,1)*(1-(U(iv,2)));
    end
    [hh,indtest] = max(test);
```

```
    Best = UtilityMaxFeas11(PUM,U,SS,PtrTrain,sum(SS(1,:)),...
        ytrTrain,0.01,0.0,Target,hyp1,hyp01,hyp2,hyp02,A12);
    ytr1 = 2*((Best(1)^2 + 3^Best(2)<10)...
        || (Best(1)*3 + Best(3)^2<10)...
        || (Best(2)*3 + Best(3)^2<10)>rand(1,1))-1;
    if (ytr1==1)
        s1 = dp(Best);
        s2 = levy(Best);
        Y1 = [Y1;s1 s2];
        PtrTrain = [PtrTrain;Best];
        ytrTrain = [ytrTrain;Y1];
        rev = 1;
        PPsame = 0;
        for tr=1:length(PP(:,1))
            if (Best(1)==PP(tr,1)) && (Best(2)==PP(tr,2))...
                    && (Best(3)==PP(tr,3))
                disp('same one')
                PPsame = tr;
                break;
            end
        end
        if PPsame~=0
            PP(PPsame,:) = [];
        end
        A12 = [A12;Best 1 s1 s2];
    else
        dp11 = dp(Best);
        levy11 = levy(Best);
        Y1 = [Y1;dp11 levy11];
        A12 = [A12;Best -1 dp11 levy11];
        rev = 0;
        PPsame = 0;
        for tr=1:length(PP(:,1))
            if (Best(1)==PP(tr,1)) && (Best(2)==PP(tr,2))...
                    && (Best(3)==PP(tr,3))
                disp('same one')
                PPsame = tr;
                break;
            end
        end
        if PPsame~=0
            PP(PPsame,:) = [];
        end
    end
end
%-------------------------------------------------------------------
```

```
    C1 = Y1;
    if rev==1
        figure(2)
        plot(C1(:,1),C1(:,2),'r.','markersize',10)
        hold on
        figure(11);
        plot3(Best(:,1),Best(:,2),Best(:,3),'r*')
        hold on
    else
        disp('black')
        figure(2)
        plot(C1(:,1),C1(:,2),'k.','markersize',10)
        hold on
        figure(11);
        plot3(Best(:,1),Best(:,2),Best(:,3),'k*')
        hold on
    end
    %-------------------------------------------------
end
hold off


function [dd,dd1] = GPclassifierFeas(AA,PP)

% Function to perform binary classification
% Input: AA - the training set
%        PP - the set of candidate solutions
% Output: dd  - all candidate solutions that are feasible
%         dd1 - all candidate solutions that are infeasible

id = [3,3];
cov = {@covSEard}; sf = 1; ell = [.2 .2 .2];              % setup the GP
hyp0.cov  = log([ell sf]);
mean = {@meanZero};                                       % m(x) = 0
hyp0.mean = [];
lik_list = {'likGauss','likErf','likLogistic'};  % allowable likelihoods
inf_list = {'infExact','infLaplace','infEP','infVB'};% poss inference algs

Ncg = 50;                              % number of conjugate gradient steps
for i=1:size(id,1)
    lik = lik_list{id(i,1)};                        % setup the likelihood
    if strcmp(lik,'likGauss')
        sn = .2; hyp0.lik = log(sn);
    else
        hyp0.lik = [];
    end
```

```
    inf = inf_list{id(i,2)};
    hyp = minimize(hyp0,'gp', -Ncg, inf, mean, cov, lik, AA(:,1:3), AA(:,4));
    [ymu, ys2] = gp(hyp, inf, mean, cov, lik,  AA(:,1:3), AA(:,4), PP);
    [nlZ] = gp(hyp, inf, mean, cov, lik, AA(:,1:3), AA(:,4));
end
d = [ymu ys2];dd = [];dd1 = [];
for i=1:length(ymu)
    if (d(i,1)+d(i,2))>=0.9
        dd = [dd;PP(i,:)];
    else
        dd1 = [dd1;PP(i,:)];
    end
end


function Best = UtilityMaxFeas11(PU,U,PP,PtrTrain,range,ytrTrain,...
epsilon,Crate,Target,hyp10,hyp01,hyp20,hyp02,A12)

% Function that uses a Genetic Algorithm (GA) to do a finegrane search for the
% optimal candidate solution
% Input: U - matrix of fitness, first row-Hy1, second-ym1.
%        PU - matrix of formulations for initial population.
%        PP - the set of candidate solutions
%        PtrTrain - the set of solutions collected so far
%        ytrTrain - the set of observations collected so far
%        range - the range of the input values
%        epsilon - parameter for the computation of pertrubations during
%        mutation step of the GA
%        Crate - crossover rate (set to 0)
%        Target - vector of target values
%        hyp10,hyp01,hyp20,hyp02 - hyperparameters of the GP models
%        A12 - the current training set (with labels)
% Output: Best - the next candidate solution to evaluate

test = zeros(length(U(:,1)),1);
for i=1:length(U(:,1))
    test(i) = U(i,1)*(1-(U(i,2)));
end
[hh,indtest] = max(test);
bestVol = hh;
Best = PU(indtest,:);
I = indtest;
id = [1,1];
cov = {@covSEard};
mean = {@meanZero};
lik_list = {'likGauss','likLaplace','likSech2','likT'};
```

```
inf_list = {'infExactNew','infLaplace','infEP','infVB'};
for i=1:size(id,1)
    lik = lik_list{id(i,1)};
    inf = inf_list{id(i,2)};
end
%-------------------------------------------------------------------------
O = InitialApproachOnline(I,PU,range,U,epsilon,Crate);
if isempty(O)==1
    return;
end
[O,SS1] = GPclassifierFeas(A12,O);
if isempty(O)==1
    return;
end
[ymu1{1}, ys21{1}] = gpCorrected(hyp10, inf, mean, cov, lik, PtrTrain,...
    ytrTrain(:,1),O);  % predict
if isempty(ymu1{1})==1 || isempty(ys21{1})==1
    [ymu1{1}, ys21{1}] = gpCorrected(hyp01, inf, mean, cov, lik,...
        PtrTrain, ytrTrain(:,1),O);  % predict
end
[ymu2{1}, ys22{1}] = gpCorrected(hyp20, inf, mean, cov, lik, PtrTrain,...
    ytrTrain(:,2),O);  % predict
if isempty(ymu2{1})==1 || isempty(ys22{1})==1
    [ymu2{1}, ys22{1}] = gpCorrected(hyp02, inf, mean, cov, lik,...
        PtrTrain, ytrTrain(:,2),O);  % predict
end
ym1 = ymu1{1};ym2 = ymu2{1};
Hy1 = ActiveDesignOnline(PtrTrain,PP,O,hyp10,hyp01);
Hy2 = ActiveDesignOnline(PtrTrain,PP,O,hyp20,hyp02);
yy = mgaFunctions([ym1 ym2],Target);
yy(:,1) = yy(:,1)/max(yy(:,1));yy(:,2) = yy(:,2)/max(yy(:,2));
U1 = [];
for i=1:length(O(:,1))
    U1(i,:) = [sqrt(Hy1(i)+Hy2(i)) sqrt(yy(i,1)^2+yy(i,2)^2)];
end
test = zeros(length(U1(:,1)),1);
for i=1:length(U1(:,1))
    test(i) = U1(i,1)*(1-(U1(i,2)));
end
[hh,indtest1] = max(test);
if hh>bestVol
    bestVol = hh;
    Best = O(indtest1,:);
    I = indtest1;
end
%-------------------------------------------------------------------------
```

```
Past = PU;New = O; PastS = U; NewS = U1;
for h=1:100
    [offsprings] = ElitistApproachOnline(I,Past,New,PastS,NewS,range,...
        epsilon,Crate);
    Po = offsprings;
    if isempty(Po)==1
        return;
    end
    [Po,SS1] = GPclassifierFeas(A12,Po);
    if isempty(Po)==1
        return;
    end
    [ymu1{1}, ys21{1}] = gpCorrected(hyp10, inf, mean, cov, lik,...
        PtrTrain, ytrTrain(:,1),Po);  % predict
    if isempty(ymu1{1})==1 || isempty(ys21{1})==1
        [ymu1{1}, ys21{1}] = gpCorrected(hyp01, inf, mean, cov, lik,...
            PtrTrain, ytrTrain(:,1),Po);  % predict
    end
    [ymu2{1}, ys22{1}] = gpCorrected(hyp20, inf, mean, cov, lik,...
        PtrTrain, ytrTrain(:,2),Po);  % predict
    if isempty(ymu2{1})==1 || isempty(ys22{1})==1
        [ymu2{1}, ys22{1}] = gpCorrected(hyp02, inf, mean, cov, lik,...
            PtrTrain, ytrTrain(:,2),Po);  % predict
    end
    ym1 = ymu1{1};ym2 = ymu2{1};
    Hy1 = ActiveDesignOnline(PtrTrain,PP,Po,hyp10,hyp01);
    Hy2 = ActiveDesignOnline(PtrTrain,PP,Po,hyp20,hyp02);
    yy = mgaFunctions([ym1 ym2],Target);
    yy(:,1) = yy(:,1)/max(yy(:,1));yy(:,2) = yy(:,2)/max(yy(:,2));
    U1 = [];
    for i=1:length(Po(:,1))
        U1(i,:) = [sqrt(Hy1(i)+Hy2(i)) sqrt(yy(i,1)^2+yy(i,2)^2)];
    end
    test = zeros(length(U1(:,1)),1);
    for i=1:length(Po(:,1))
        test(i) = U1(i,1)*(1-(U1(i,2)));
    end
    [hh,indtest2] = max(test);
    if hh>bestVol
        bestVol = hh;
        Best = Po(indtest2,:);
        I = indtest2;
    end
    Past = New; New = Po;PastS = NewS;NewS = U1;
end
```

Appendix C

Matlab code for categorisation of formulations as discussed in chapter 4

The code presented in Appendix C is the one employed in the classification of data as discussed in chapter 4. Matlab built in functions 'knnclassify' and 'svmclassify' were used to obtain performance results for kNN and SVM classifiers respectively. For visualisation, functions from Matlab toolbox provided by [69] were used.

The function 'classifyIsomapTest31' was run in conjunction with the SIsomap Matlab code provided by [65].

```
function [OverallRate,S] = classifyIsomapTest31(data,alpha,K)

% Function that uses SIsomap based approach for classification applied to
% the first dataset as discussed in chapter 4.
% Input: data - the 166 x 6 data matrix where the rows are individual
%               formulations and the columns are proportions of each
%               individual ingredient
%        alpha - parameter of the SIsomap algorithm influencing the inter
%                class overlap, chosen from (0,1) interval
%        K - number of neighbours for the construction of the neighbourhood
%        graph
% Output: OverallRate - mean (over 10 runs) performance, as a percentage of
%         correctly classified formulations
```

```
%         S – standard deviation of the result

for p1=1:10
    c1 = cvpartition(data(:,8),'kfold',10);
    for p=1:10
        idxTrain1 = training(c1,p);
        trainIt1 = [];testIt1 = [];
        for jj=1:length(data(:,1))
            if idxTrain1(jj)==1
                trainIt1 = vertcat(trainIt1,data(jj,:));
            end
        end
        for ll=1:length(data(:,1))
            if idxTrain1(ll)==0
                testIt1 = vertcat(testIt1,data(ll,:));
            end
        end
        data1 = trainIt1;
        D2 = [];w1 = [];
        for m=1:5
            b2 = DataSort3(data1,m); % working out class imbalance
            w1 = [w1 length(b2)];
            D2 = [D2;b2];
        end
        aa = max(w1);
        length(D2);
        I = [];w2 = [];
        for ii=1:5
            ss = syntData4(D2,ii,aa-w1(ii),1.3); % equating the classes
            w2 = [w2 length(ss)];
            I = [I;ss];
        end
        good = 0; bad = 0;
        D1 = [];
        for m1=1:5
            b1 = DataSort1(data1,m1,1,2); % calculating cluster centres
            D1 = [D1;b1];
        end
        r1 = sIsomap(I,alpha,K);
        net1 = newgrnn(I(:,1:6)',r1(1:3,:),2);
        V = sim(net1,testIt1(:,1:6)');
        classV = knnclassify(V',r1(1:3,:)',r1(4,:), 35);
        fail = [];
        for n=1:length(testIt1(:,1))
            if classV(n)==testIt1(n,8)
                good = good + 1;
```

```
            else
                bad = bad + 1;
                fail = [fail;testIt1(classV(n),:)];
            end
        end
        %-------------------------------------------------------------------
        if isempty(fail)==0
            for f=1:length(fail(:,1))
                lowestD = 1000;
                for f1=1:5
                    total = 0;
                    for a=1:6
                        total = total + (fail(f,a)-D1(f1,a))^2;
                    end
                    if sqrt(total) < lowestD
                        lowestD = sqrt(total);
                        l = f1;
                    end
                end
                lowestTest(f) = l;
            end
            for n1=1:length(fail(:,1))
                if lowestTest(n1)==fail(n1,8)
                    good = good + 1;

                end
            end
        end
        %-------------------------------------------------------------------
        rate(p) = good/length(testIt1(:,1));
    end
    OverallRate2(p1) = sum(rate)/10;
end
S = std(OverallRate2);
OverallRate = sum(OverallRate2)/10;


function [Category] = DataSort3(data,m)

% Function to sort data into sets according to the class label
% Input: data - the 166 x 6 data matrix where the rows are individual
%               formulations and the columns are proportions of each
%               individual ingredient
%        m - class label (on interger scale, 1,2,...,5)
% Output: Category - matrix of formulations with class label m
```

```
Category = [];
for y=1:length(data(:,1))
    if data(y,8)==m
        Category = vertcat(Category,data(y,:));
    end
end


function [C] = DataSort1(data,m)

% Function to compute cluster centres
% Input: data – the 166 x 6 data matrix where the rows are individual
%               formulations and the columns are proportions of each
%               individual ingredient
%        m – class label (on interger scale, 1,2,...,5)
% Output: C – coordinates of the cluster center for all formulations with
%         class label m

Category = [];
for y=1:length(data(:,1))
    if data(y,8)==m
        Category = vertcat(Category,data(y,:));
    end
end
[IDX,C,sumd,D] = kmeans(Category,1);


function [OverallRate,S] = classifyIsomapTree(data,Q)

% Function that uses decision tree algorithm for classification applied to
% the first dataset as discussed in chapter 4.
% Input: data – the 166 x 6 data matrix where the rows are individual
%               formulations and the columns are proportions of each
%               individual ingredient
%        Q – is the number such that impure nodes must have Q or more
%            observations to be split
% Output: OverallRate – mean (over 10 runs) performance, as a percentage of
%         correctly classified formulations
%         S – standard deviation of the result

for p1=1:10
    c1 = cvpartition(data(:,8),'kfold',10);
    for p=1:10
        idxTrain1 = training(c1,p);
        trainIt1 = [];testIt1 = [];
```

```
for jj=1:length(data(:,1))
    if idxTrain1(jj)==1
        trainIt1 = vertcat(trainIt1,data(jj,:));
    end
end
for ll=1:length(data(:,1))
    if idxTrain1(ll)==0
        testIt1 = vertcat(testIt1,data(ll,:));
    end
end
data1 = trainIt1;
good = 0; bad = 0;
D1 = [];
for m1=1:5
    b1 = DataSort1(data1,m1); % calculating cluster centres
    D1 = [D1;b1];
end
% Start with a large tree.
t = treefit(data1(:,1:6),data1(:,8),'splitmin',Q);
% Find the minimum-cost tree.
[c,s,n,best] = treetest(t,'cross',data1(:,1:6),data1(:,8));
tmin = treeprune(t,'level',best);
sfit = treeval(tmin,testIt1(:,1:6));     % Find assigned class numbers
classV = round(sfit);
fail = [];
for n=1:length(testIt1(:,1))
    if classV(n)==testIt1(n,8)
        good = good + 1;
    else
        bad = bad + 1;
        fail = [fail;testIt1(n,:)];
    end
end
%-------------------------------------------------------------------------
if isempty(fail)==0
    for f=1:length(fail(:,1))
        lowestD = 1000;
        for f1=1:5
            total = 0;
            for a=1:6
                total = total + (fail(f,a)-D1(f1,a))^2;
            end
            if sqrt(total) < lowestD
                lowestD = sqrt(total);
                l = f1;
            end
```

```
                end
                lowestTest(f) = l;
            end
            for n1=1:length(fail(:,1))
                if lowestTest(n1)==fail(n1,8)
                    good = good + 1;

                end
            end
        end
        %-------------------------------------------------------------------------
        rate(p) = good/length(testIt1(:,1));
    end
    OverallRate2(p1) = sum(rate)/10;
end
S = std(OverallRate2);
OverallRate = sum(OverallRate2)/10;


function y = classifyNN(D)

% Function that uses Neural Network algorithm for classification and
% applied to
% the first dataset as discussed in chapter 4.
% Input: data – the 166 x 6 data matrix where the rows are individual
%               formulations and the columns are proportions of each
%               individual ingredient
% Output: y – mean (over 10 runs) performance, as a percentage of
%         correctly classified formulations
%         S – standard deviation of the result

for h=1:10
    c = cvpartition(D(:,8),'kfold',10);
    good = 0; bad = 0;

    for i=1:10
        idxTrain = training(c,i);
        trainIt = [];testIt = [];
        for j=1:length(D(:,1))
            if idxTrain(j)==1
                trainIt = vertcat(trainIt,D(j,:));
            end
        end
        for l=1:length(D(:,1))
            if idxTrain(l)==0
                testIt = vertcat(testIt,D(l,:));
```

```
        end
    end
    D1 = [];
    for m1=1:3
        b1 = DataSort1(trainIt,m1,1,2); % calculating cluster centres
        D1 = [D1;b1];
    end
    net2 = newff(trainIt(:,1:6)',trainIt(:,8)',[5]);% create neural network
    net2 = train(net2,trainIt(:,1:6)',trainIt(:,8)');% train network
    out = round(sim(net2,testIt(:,1:6)'));
    fail = [];
    for n=1:length(testIt(:,1))
        if out(n)==testIt(n,8)
            good = good + 1;
        else
            bad = bad + 1;
            fail = [fail;testIt(n,:)];
        end
    end
    g=0;
    %-------------------------------------------------------------------
    if isempty(fail)==0
        for f=1:length(fail(:,1))
            lowestD = 1000;
            for f1=1:3
                total = 0;
                for a=1:6
                    total = total + (fail(f,a)-D1(f1,a))^2;
                end
                if sqrt(total) < lowestD
                    lowestD = sqrt(total);
                    l = f1;
                end
            end
            lowestTest(f) = l;
        end
        for n1=1:length(fail(:,1))
            if lowestTest(n1)==fail(n1,8)
                good = good + 1;
                g=g+1;
            end
        end
    end
    rate(i) = good/length(testIt(:,1));
    good = 0; bad = 0;
end
```

```
    y(h) =sum(rate)/10;
end
S = std(y);
y1 = sum(y)/10;
```

Bibliography

[1] Costa, R., Moggridge, G.D, Saraiva, P.M. (2006). Chemical Product Engineering - An Emerging Paradigm within Chemical Engineering. *AIChEJ*, **52**(6), pp. 1976-1986.

[2] Vemuri, V.R. (1993). Main problems and issues in neural networks application. *Artificial Neural Networks and Expert Systems. Proceedings., First New Zealand International Two-Stream Conference on*, 24-26 Nov 1993, pp. 226.

[3] Bellman, R.E. (1961). *Adaptive Control Processes*. Princeton University Press.

[4] Aman, K., Kocijan, J. (2007). Application of Gaussian processes for black-box modelling of biosystems. *ISA Transactions*, **46**(4), pp. 443-457.

[5] Xia, W., Luo, B., Liao, X. (2010). An enhanced global optimization method based on Gaussian process and its application of warpage control in injection molding. *IEEE ICIA conference*, pp. 970-975.

[6] Villemonteix, J., Vazquez, E., Walter, E. (2009). An informational approach to the global optimization of expensive-to-evaluate functions. *Journal of Global Optimization*, **44**(4), pp. 509-534.

[7] Osborne, M.A., Garnett, R., Roberts, S.J. (2009). Gaussian processes for global optimization. *In: Learning and Intelligent Optimization Conference*, **3**.

[8] Huang, D., Allen, T.T., Notz, W.I., Zeng, N. (2006). Global Optimization of Stochastic Black-Box Systems via Sequential Kriging Meta-Models. *Journal of Global Optimization*, **34**(3), pp. 441-466.

[9] Frean, M., Boyle, P., (2008). Using Gaussian Processes to Optimize Expensive Functions. *In:* Wobcke, W., Zhang, M. eds. *Proceedings of the 21st Australasian Joint Conference on Artificial Intelligence: Advances in Artificial Intelligence (AI '08)*, Springer-Verlag, Berlin, Heidelberg, pp. 258-267.

[10] Knowles, J. (2006). ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, **10**(1), pp. 50-66.

[11] Tang, Q., Lau, Y.B., Hu, S., Yan, W., Yang, Y., Chen, T. (2010). Response surface methodology using Gaussian processes: Towards optimizing the trans-stilbene epoxidation over Co2+-NaX catalysts. *Chemical Engineering Journal*, **156**(2), pp. 423-431.

[12] McMullen, J.P., Stone, M.T., Buchwald, S.L., Jensen, K.F. (2010). An Integrated Microreactor System for Self-Optimization of a Heck Reaction: From Micro- to Mesoscale Flow Systems. *Angewandte Chemie International Edition*, **49**, pp. 7076-7080.

[13] Nelder, J.A., Mead, R. (1965). A simplex method for function minimization. *Computer Journal*, **7**, pp. 308-313.

[14] Vlachos, M., Domeniconi, C., Gunopulos, D., Kollios, G., Koudas, N. (2002). Non-linear dimensionality reduction techniques for classification and visualization. *In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '02)*, ACM, New York, pp. 645-651.

[15] Perlack, R.D., Wright, L.L., Turhollow, A.F., Graham, R.L., Stokes, B.J., Erbach, D.C. (2005). Biomass as feedstocks for a bioenergy and bioproducts industry: the technical feasibility of a billion-ton annual supply.

[16] Graham, R.L. (2007). Forecasting the magnitude of sustainable biofeedstock supplies: the challenges and the rewards. *Biofuels, Bioproducts and Biorefining*, **1**(4), pp. 255-263.

[17] Fernando, S., Adhikari, S., Chandrapal, C., Murali, N. (2006). Biorefineries: current status, challenges, and future direction. *Energy and Fuels*, **20**, 1727-1737.

[18] Smith, W. (2007). Mapping the development of UK biorefinery complexes. UK National Non Food Crops Centre (NNFCC).

[19] FitzPatrick, M., Champagne, P., Cunningham, M. F., Whitney, R.A. (2010). A biorefinery processing perspective: Treatment of lignocellulosic materials for the production of value-added products. *Bioresource Technology*, **101**(23), pp. 8915-8922.

[20] Marshall, A.-L., Alaimo, P.J. (2010). Useful Products from Complex Starting Materials: Common Chemicals from Biomass Feedstocks. *Chemistry*, **16**(17), pp. 4970-4980.

[21] Lee, H.K.H., Gramacy, R.B., Linkletter, C., Gray, G.A. (2011). Optimization subject to hidden constraints via statistical emulation. *Pacific Journal Of Optimization*, **7**(3), pp. 467-478.

[22] Mayer, A.S., Kelley, C.T., Miller, C.T. (2002). Optimal design for problems involving flow and transport phenomena in saturated subsurface systems. *Advances in Water Resources*, **25**, pp. 1233-1256.

[23] Gelman, A., Carlin, J.B., Stern, H.S., Rubin, D.B. (1995). *Bayesian Data Analysis*. London: Chapman and Hall.

[24] Breiman, L. (2001). Random Forests. *Machine Learning*, **45**, pp. 5-32.

[25] Finkel, D.E., Kelley, C.T. (2009). Convergence Analysis of Sampling Methods for Perturbed Lipschitz Functions. *Pacific Journal of Optimization*, **5**, pp. 339-350.

[26] McLachlan, G., Peel, D., (2000). *Finite Mixture Models*. Hoboken, NJ: John Wiley Sons, Inc.

[27] Abramowitz, M., Stegun, I.A. (1965). *Handbook of Mathematical Functions*. Dover, New York, pp. 84-85.

[28] Cox, D., John, S. (1997). A statistical method for global optimization. *In:* Alexandrow, N., Hussaini, M., eds. *Multidisciplinary design optimization: State of the art*, Philadelphia: SIAM, pp. 315-329.

[29] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, **6**(2), pp. 182-197.

[30] El-Beltagy, M.A., Keane, A.J. (2001). Evolutionary optimization for computationally expensive problems using Gaussian processes. *In: Proceedings of the International Conference on Artificial Intelligence*, CSREA Press, pp. 708-714.

[31] Guestrin, C., Krause, A., Singh, A. (2005). Near-optimal Sensor Placements in Gaussian Processes. *In: International Conference on Machine Learning (ICML)*.

[32] Harrington, J. (1965). The desirability function. *Industrial Quality Control*, **21**(10), pp. 494-498.

[33] Jones, D., Schonlau, M., Welch, W. (1998). Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, **13**, pp. 455-492.

[34] Peremezhney, N., Connaughton, C., Unali, G., Hines, E., Lapkin, A.A., (2012). Application of dimensionality reduction to visualisation of high-throughput data and building of a classification model in formulated consumer product design. *Chemical Engineering Research and Design*, **90**(12), pp. 2179-2185.

[35] Rasmussen, C.E., Williams C.K.I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

[36] Srinivas, N., Krause, A., Kakade, S., Seeger, M. (2010). Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. *International Conference on Machine Learning*, pp. 1015-1022.

[37] Wenzel, S., Straatmann, S., Kwiatkowski, L., Schmelzer, P., Kunert, J. (2010). A Novel Multi-Objective Target Value Optimization Approach. *In:* Locarek-Junge, H., Weihs, C., eds. *Classification as a Tool for Research: Studies in Classification, Data Analysis, and Knowledge Organization*, pp. 801-809.

[38] Zitzler, E., Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms A comparative case study. *In:* Eiben, A.E., Back, T., Schoenauer, M., Schwefel, H.P., eds. *Parallel Problem Solving From Nature, V*, Berlin: Springer-Verlag, pp. 292-301.

[39] Minka, T.P. (2001). *A Family of Algorithms for Approximate Bayesian Inference*. Ph.D. thesis. Massachusetts Institute of Technology.

[40] Emmerich, M., Giannakoglou, K., Naujoks, B. (2006). Single and multiobjective evolutionary optimization assisted by Gaussian random field metamodels. *IEEE Transactions on Evolutionary Computation*, **10**(4), pp. 421-439.

[41] Liu, B., Zhang, Q., Gielen, G. (2013). A Gaussian Process Surrogate Model Assisted Evolutionary Algorithm for Medium Scale Expensive Optimization Problems. *IEEE Transactions on Evolutionary Computation*, **PP**(99).

[42] Ackley, D., H. (1987). *A connectionist machine for genetic hillclimbing*. Boston: Kluwer Academic Publishers.

[43] Momin, J., Xin-She, Y. (2013). A literature survey of benchmark functions for global optimization problems. *Int. Journal of Mathematical Modelling and Numerical Optimisation*, **4**(2), pp. 150-194.

[44] Laguna, M., Marti, R. (2002). Experimental Testing of Advanced Scatter Search Designs for Global Optimization of Multimodal Functions. Retrieved June 2013, from http://www.uv.es/rmarti/paper/docs/global1.pdf.

[45] Dixon, L., C., W., Price, R., C. (1989). The Truncated Newton Method for Sparse Unconstrained Optimisation Using Automatic Differentiation. *Journal of Optimisation Theory and Applications*, **60**(2), pp. 261-275.

[46] McKay, M., D., Beckman, R., J., Conover, W., J. (1979). A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics*, **21**(2), pp. 239-245.

[47] Mathworks.com. (2014). *Yi Cao - File Exchange - MATLAB Central*. [online] Available at:
http://www.mathworks.com/matlabcentral/fileexchange/authors/22524
[Accessed: 1 Apr 2014].

[48] Painter, P.C., Coleman, M.M. (1997). *Fundamentals of polymer science : an introductory text*. Lancaster, Pa.: Technomic Pub. Co.

[49] McCrum, N.G., Buckley, C.P., Bucknall, C.B. (1997). *Principles of polymer engineering*. Oxford, New York: Oxford University Press.

[50] Clayden, J., Greeves, N., Warren, S. (2000). *Organic chemistry*. Oxford University Press, pp. 1450-1466.

[51] Kiparissides, C. (2006). Challenges in Particulate Polymerization Reactor Modeling and Optimization: A Population Balance Perspective. *Journal of Process Control*, **16**, pp. 205-224.

[52] Chern, C.S. (2006). Emulsion polymerization mechanisms and kinetics. *Progress in Polymer Science*, **31**(5), pp. 443-486.

[53] Li, B.-G., Brooks, B.W. (1993). Prediction of the average number of radicals per particle for emulsion polymerization. J. *Journal of Polymer Science, Part A: Polymer Chemistry*, **31**(9), pp. 2397-2402.

[54] Zubov, A., Pokorny, J., Kosek, J. (2012). Styrene-butadiene Rubber (SBR) Production by Emulsion Polymerization: Dynamic Modeling and Intensification of the Process. *Chemical Engineering Journal*, **207-208**, pp. 414-420.

[55] Gugliotta, L.M., Arzamendi, G., Asua, J.M. (1995). Choice of Monomer Partition Model in Mathematical Modeling of Emulsion Copolymerization Systems. *Journal of Applied Polymer Science*, **55**, pp. 1017-1039.

[56] Urretabizkaia, A., Asua, J.M. (1994). High Solids Content Emulsion Terpolymerization of Vinyl-Acetate, Methyl-Methacrylate and Butyl Acrylate. 1. Kinetics. *Journal of Polymer Science Part A: Polymer Chemistry*, **32**, pp. 1761-1778.

[57] Du, M., Zhang, S., Wang, H. (2009). Supervised Isomap for Plant Leaf Image Classification. *Emerging Intelligent Computing Technology and Applications. With Aspects of Artificial Intelligence Lecture Notes in Computer Science*, **5755**, pp. 627-634.

[58] Vieira, A., Ribeiro, B., Chen, N. (2012). Credit Scoring for SME Using a Manifold Supervised Learning Algorithm. *Intelligent Data Engineering and Automated Learning - IDEAL 2012 Lecture Notes in Computer Science*, **7435**, pp. 763-770.

[59] Wang, S., Li, H. (2013). A facial expression recognition method based on supervised Isomap. *World Congress on Medical Physics and Biomedical Engineering May 26-31, 2012, Beijing, China IFMBE Proceedings*, **39**, pp. 473-476.

[60] Zadeh, L.A. (1965). Fuzzy sets. *Information and Control*, **8**(3), pp. 338-353.

[61] Jolliffe, I.T. (2002). *Principal Component Analysis*. Springer, New York.

[62] Borg, I., Groenen, P. (2005). *Modern Multidimensional Scaling: Theory and Application*. Springer, New York, NY.

[63] Shlens, J. (2005). *A Tutorial on Principal Component Analysis*. Institute for Nonlinear Science, UCSD.

[64] Roweis, S.T., Saul, L.K. (2000). Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, **290**, pp. 2323-2326.

[65] Tenenbaum, J.B., Silva, V.De., Langford, J.C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, **290**, pp. 2319-2323.

[66] Geng, X., De-Chuan, Z., Zhi-Hua, Z. (2005). Supervised nonlinear dimensionality reduction for visualization and classification. *IEEE Transactions on systems, man, and cybernetics, Part B: Cybernetics*, **35**, pp. 1098-1107.

[67] Fisher, R. (1936). The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics*, **7**(2), pp. 179-188.

[68] Saul, L.K., Roweis, S.T. (2000). *An introduction to locally linear embedding.* AT & T Labs and Gatsby Computational Neuroscience Unit.

[69] Maaten, L. (2010). *Matlab Toolbox for Dimensionality Reduction.* [Online]. (URL http://homepage.tudelft.nl/19j49/Matlab_Toolbox_for_Dimensionality_Reduction.html). (Accessed 5 April 2011).

[70] Tenenbaum, J.B., Silva, V.De., Langford, J.C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science.* **290**, pp. 2319-2323.

[71] Wasserman, P.D. (1993). *Advanced Methods in Neural Computing.* Van Nostrand Reinhold, New York.

[72] Ho, T.K. (1998). Nearest Neighbours in random subspaces. *Lecture notes in computer Science: Advances in Pattern Recognition.* **1451**, pp. 640-948.

[73] He, Y., Ding, X., Ma, L. (2012). Supervised Manifold Learning for NIR Modeling of Cigarette Brand Identification. *Advanced Materials Research*, **457-458**, pp. 1258-1263.

[74] Wang, S., Yang, H., Li, H. (2013). Facial Expression Recognition Based on Incremental Isomap with Expression Weighted Distance. *Journal of Computers*, **8**(8), pp. 2051-2-58.

[75] Seber, G.A.F. (1984). *Multivariate Observations.* John Wiley & Sons, Inc, Hoboken, NJ.

[76] Rosenblatt, F. (1961). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms.* Spartan Books, Washington DC.

[77] Vapnik, V. (1998). *Statistical Learning Theory.* John Wiley and Sons, New York.

[78] Breiman, L., Friedman, J., Olshen, R., Stone, C. (1984). *Classification and Regression Trees.* CRC Press, Boca Raton, FL.

[79] Kaufman, L., Rousseeuw, P. (1990). *Finding Groups in Data: an Introduction to Cluster Analysis*. Wiley, New York.

[80] Theodoridis, S., Konstantinos, K. (2008). *Pattern Recognition*. 4th ed. Academic Press.

[81] Nie, F., Xu, D., Li X., Xiang, S. (2011). Semi-Supervised Dimensionality Reduction and Classification through Virtual Label Regression. *IEEE Transactions on systems, man, and cybernetics, Part B: Cybernetics*, **41**(3), pp. 675-85.

[82] Bartosz, A., Grzybowski, K.J., Bishop, M., Kowalczyk, B., Wilmer, C.E. (2009). The wired universe of organic chemistry. *Nature Chemistry*, **1**, pp. 31-36.

[83] Todd, M.H. (2005). Computer-aided organic synthesis. *Chemical Society Review*, **34**, pp. 247-266.

[84] Chen, J.H., Baldi, P. (2009). No electron left behind: a rule-based expert system to predict chemical reactions and reaction mechanisms. *Chemical Information and Modelling*, **49**(9), pp. 2034-2043.

[85] Socorro, I.M., Goodman, J.M. (2006). The ROBIA program for predicting organic reactivity. *Journal of Chemical Information and Modeling*, **46**(2), pp. 606-614.

[86] Jorgensen, W.L., Laird, E.R., Gushurst, A.J., Fleischer, J.M., Gothe, S.A., Helson, H.E., Paderes, G.D., Sinclair, S. (1990). CAMEO: A program for the logical prediction of the products of organic reactions. *Pure and Applied Chemistry*, **62**, pp. 1921-1932.

[87] Satoh, H., Funatsu, K. (1995). SOPHIA, a Knowledge Base Guided Reaction Prediction System - Utilization of a Knowledge Base Derived from a Reaction Database. *Journal of Chemical Information and Computer Science*, **35**(1), pp. 34-44.

[88] Law, J., Zsoldos, Z., Simon, A., Reid, D., Liu, Y., Khew, S.Y., Johnson, A.P., Major, S., Wade, R.A., Ando, H.Y. (2009). Route Designer: A Retrosynthetic Analysis Tool Utilizing Automated Retrosynthetic Rule Generation. *Chemical Information and Modelling*, **49**, pp. 593-602.

[89] Nowak, G., Fic, G. (2010). Search for Complexity Generating Chemical Transformations by Combining Connectivity Analysis and Cascade Transformation Patterns. *Chemical Information and Modelling*, **50**, pp. 1369-1377.

[90] Frazer, A., Hensler, J.G. (1999). Chapter 13: Serotonin Receptors. In Siegel, G.J., Agranoff, B.W., Albers, R.W., Fisher, S.K., Uhler, M.D., editors. *Basic Neurochemistry: Molecular, Cellular, and Medical Aspects*, Philadelphia: Lippincott-Raven. pp. 263-292.

[91] Rsc.org. (2014). *Methods in Organic Synthesis Home*. [online] Available at: http://www.rsc.org/Publishing/CurrentAwareness/MOS/ [Accessed: 28 Mar 2014].

[92] Lee, J.W., Kim, H.U., Choi, S., Yi, J., Lee, S.Y. (2011). Microbial production of building block chemicals and polymers. *Current Opinion in Biotechnology*, **22**, pp. 1-10.

[93] Rozenman, M.M., McNaughton, B.R., Liu, D.R. (2007). Solving chemical problems through the application of evolutionary principles. *Current Opinion in Chemical Biology*, **11**, pp. 259-268.

[94] Hunt, R.A.R., Otto, S. (2011). Dynamic combinatorial libraries: new opportunities in systems chemistry. *Chemical Communications*, **47**, pp. 847-858.

[95] Moulin, E., Cormos, G., Giuseppone, N. (2012). Dynamic combinatorial chemistry as a tool for the design of functional materials and devices. *Chemical Society Reviews*, **41**(3), pp. 1031-49

[96] Yadav, V.G., Stephanopoulos, G. (2010). Reevaluating synthesis by biology. *Current Opinion in Microbiology*, **13**, pp. 371-376.

[97] Leemhuis, H., Kelly, R.M., Dijkhuizen, L. (2009). Critical Review Directed Evolution of Enzymes: Library Screening Strategies. *Life*, **61**(3), pp. 222-228.

[98] Johannes, T.W., Zhao, H. (2006). Directed evolution of enzymes and biosynthetic pathways. *Current Opinion in Microbiology*, **9**, pp. 261-267.

[99] Netsci.org. (2014). *Combinatorial Chemistry: A Strategy for the Future*. [online] Available at: http://www.netsci.org/Science/Combichem/feature02.html [Accessed: 28 Mar 2014].

[100] Zhao, H., Chockalingam, K., Chen, Z. (2002). Directed evolution of enzymes and pathways for industrial Biocatalysis. *Current Opinion in Biotechnology*, **13**, pp. 104-110.

[101] Corbett, P.T., Leclaire,J., Vial, L., West, K.R., Wietor, J.-L., Sanders, J.K.M., Otto, S. (2006). Dynamic Combinatorial Chemistry. *Chemical Reviews*, **106**, pp. 3652-3711.

[102] Miller, B.L. (2009). *Dynamic Combinatorial Chemistry: An Introduction, in Dynamic Combinatorial Chemistry: In Drug Discovery, Bioorganic Chemistry, and Materials Science.* John Wiley & Sons, Inc., Hoboken, NJ, USA.

[103] Gartner, Z.J. (2006). Evolutionary approaches for the discovery of functional synthetic small molecules. *Pure Applied Chemistry*, **78**, pp. 1-14.

[104] Paula, J.D., Atkins, P.W. (2002). *Atkins' Physical Chemistry.* 7th ed. OUP Oxford.

[105] Hanai, T., Atsumi, S., Liao, J.C. (2007). Engineered synthetic pathway for isopropanol production in Escherichia coli. *Applied and Environmental Microbiology*, **73**(24), pp. 7814-7818.

[106] Papa, A.J. (2005). *Ullmann's Encyclopedia of Industrial Chemistry.* Weinheim: Wiley-VCH.

[107] Jung, Y.K., Kim, T.Y., Park, S.J., Lee, S.Y. (2010). Metabolic engineering of Escherichia coli for the production of polylactic acid and its copolymers. *Biotechnology and Bioengineering*, **105**(1), pp. 161-171.

[108] Nasr, G., Petit, E., Supuran, C.T., Winum, J.Y., Barboiu, M. (2009). Carbonic anhydrase II-induced selection of inhibitors from a dynamic combinatorial library of Schiff's bases. *Bioorganic & Medicinal Chemistry Letters*, **19**(21), pp. 6014-6017.

[109] Nasr, G., Petit, E., Vullo, D., Winum, J.Y., Supuran, C.T., Barboiu, M. (2009). Carbonic anhydrase-encoded dynamic constitutional libraries: toward the discovery of isozyme-specific inhibitors. *Journal of Medicinal Chemistry*, **42**(15), pp. 4853-4859.

[110] Gareiss, P.C., Sobczak, K., McNaughton, B.R., Palde, P.B., Thornton, C.A., Miller, B.L. (2008). Dynamic Combinatorial Selection of Molecules Capable of Inhibiting the (CUG) Repeat RNAMBNL1 Interaction In Vitro: Discovery of Lead Compounds Targeting Myotonic Dystrophy (DM1). *Journal of the American Chemical Society*, **130**, pp. 16254-16261.

[111] Bugaut, A., Jantos, K., Wietor, J.L., Rodriguez, R., Sanders, J.K.M., Balasubramanian, S. (2008). Exploring the Differential Recognition of DNA G-Quadruplex Targets by Small Molecules Using Dynamic Combinatorial Chemistry. *Angewandte Chemie International Edition*, **47**(14), pp. 2677-2680.